Name

groff_mom − modern macros for document composition with GNU *roff*

Synopsis

groff −mom [*option* . . .] [*file* . . .]
groff −m mom [*option* . . .] [*file* . . .]

Description

mom is a macro set for *groff*, designed primarily to prepare documents for PDF and PostScript output. *mom* provides macros in two categories: typesetting and document processing. The former provide access to *groff*'s typesetting capabilities in ways that are simpler to master than *groff*'s requests and escape sequences. The latter provide highly customizable markup tags that allow the user to design and output professional-looking documents with a minimum of typesetting intervention.

Files processed with *pdfmom*(1) produce PDF documents. The documents include a PDF outline that appears in the navigation pane panel of document viewers, and may contain clickable internal and external links.

Normally. *groff*'s native PDF driver, *gropdf*(1), is used to generate the output. When *pdfmom* is given the "**−T ps**" option, it still produces PDF, but processing is delegated to *pdfroff*, which uses *groff*'s PostScript driver, *grops*(1). Not all PDF features are available when **−T ps** is given; its primary use is to allow processing of files with embedded PostScript images.

Files processed with **groff −mom** (or **−m mom**) format for the device specified with the **−T** option. (In this installation, **ps** is the default output device.)

mom comes with her own comprehensive documentation in HTML. A PDF manual, "Producing PDFs with *groff* and *mom*", discusses preparation of PDF documents with *mom* in detail.

Files

/usr/local/share/groff/1.23.0/tmac/mom.tmac
is a wrapper enabling the package to be loaded with "**groff −m mom**".

/usr/local/share/groff/1.23.0/tmac/om.tmac
implements the package.

/usr/local/share/doc/groff−1.23.0/html/mom/toc.html
is the entry point to the HTML documentation.

/usr/local/share/doc/groff−1.23.0/pdf/mom−pdf.pdf
is "Producing PDFs with *groff* and *mom*", by Deri James and Peter Schaffter.

/usr/local/share/doc/groff−1.23.0/examples/mom/.mom*
are examples of *mom* usage.

Reference

Escape sequences

***[<*colorname*>]**
begin using an initialized colour inline

[BCK** *n]**
move backward in a line

***[BOLDER]**
invoke pseudo bold inline (related to macro **.SETBOLDER**)

***[BOLDERX]**
off pseudo bold inline (related to macro **.SETBOLDER**)

[BU** *n]**
move characters pairs closer together inline (related to macro **.KERN**)

***[COND]**
invoke pseudo condensing inline (related to macro **.CONDENSE**)

***[CONDX]**
off pseudo condensing inline (related to macro **.CONDENSE**)

[CONDSUP]**...[CONDSUPX]**
pseudo-condensed superscript

[DOWN** *n]**
temporarily move downward in a line

***[EN–MARK]**
mark initial line of a range of line numbers (for use with line numbered endnotes)

***[EXT]**
invoke pseudo extending inline (related to macro **.EXTEND**)

***[EXTX]**
off pseudo condensing inline (related to macro **.EXTEND**)

[EXTSUP]**...[EXTSUPX]**
pseudo extended superscript

[FU** *n]**
move characters pairs further apart inline (related to macro **.KERN**)

[FWD** *n]**
move forward in a line

***[LEADER]**
insert leaders at the end of a line

***[RULE]**
draw a full measure rule

[SIZE** *n]**
change the point size inline (related to macro **.PT_SIZE**)

***[SLANT]**
invoke pseudo italic inline (related to macro **.SETSLANT**)

***[SLANTX]**
off pseudo italic inline (related to macro **.SETSLANT**)

[ST<n***>]**...***[ST<***n***>X]**
string tabs (mark tab positions inline)

[SUP]**...[SUPX]**
superscript

***[TB+]**
inline escape for **.TN** (*Tab Next*)

[UL]**...[ULX]**
invoke underlining inline (fixed width fonts only)

[UP** *n]**
temporarily move upward in a line

Macros

.AUTOLEAD
set the linespacing relative to the point size

.B_MARGIN
set a bottom margin

.BR break a justified line

.CENTER
set line-by-line quad centre

.CONDENSE
set the amount to pseudo condense

.EL break a line without advancing on the page

.EXTEND
set the amount to pseudo extend

.FALLBACK_FONT
establish a fallback font (for missing fonts)

.FAM alias to **.FAMILY**

.FAMILY *<family>*
set the *family type*

.FT set the font style (roman, italic, etc.)

.HI [*<measure>* **]**
hanging indent

.HY automatic hyphenation on/off

.HY_SET
set automatic hyphenation parameters

.IB [*<left measure>* *<right measure>* **]**
indent both

.IBX [CLEAR]
exit indent both

.IL [*<measure>* **]**
indent left

.ILX [CLEAR]
exit indent left

.IQ [CLEAR]
quit any/all indents

.IR [*<measure>* **]**
indent right

.IRX [CLEAR]
exit indent right

.JUSTIFY
justify text to both margins

.KERN
automatic character pair kerning on/off

.L_MARGIN
set a left margin (page offset)

.LEFT set line-by-line quad left

.LL set a line length

.LS set a linespacing (leading)

.PAGE set explicit page dimensions and margins

.PAGEWIDTH
set a custom page width

.PAGELENGTH
set a custom page length

.PAPER *<paper_type>*
set common paper sizes (letter, A4, etc)

.PT_SIZE
set the point size

.QUAD
"justify" text left, centre, or right

.R_MARGIN
set a right margin

.RIGHT
set line-by-line quad right

.SETBOLDER
set the amount of emboldening

.SETSLANT
set the degree of slant

.SPREAD
force justify a line

.SS set the sentence space size

.T_MARGIN
set a top margin

.TI [*<measure>* **]**
temporary left indent

.WS set the minimum word space size

Documentation of details

Details of inline escape sequences in alphabetical order

***[**<*colorname*>**]**
begin using an initialized colour inline

[BCK** *n]**
move backward in a line

***[BOLDER]**
***[BOLDERX]**
Emboldening on/off

***[BOLDER]** begins emboldening type. ***[BOLDERX]** turns the feature off. Both are inline escape sequences; therefore, they should not appear as separate lines, but rather be embedded in text lines, like this:

```
Not \*[BOLDER]everything\*[BOLDERX] is as it seems.
```

Alternatively, if you wanted the whole line emboldened, you should do

```
\*[BOLDER]Not everything is as it seems.\*[BOLDERX]
```

Once ***[BOLDER]** is invoked, it remains in effect until turned off.

Note: If you're using the document processing macros with **.PRINTSTYLE TYPEWRITE**, *mom* ignores ***[BOLDER]** requests.

[BU** *n]**
move characters pairs closer together inline (related to macro **.KERN**)

***[COND]**
***[CONDX]**
Pseudo-condensing on/off

***[COND]** begins pseudo-condensing type. ***[CONDX]** turns the feature off. Both are inline escape sequences; therefore, they should not appear as separate lines, but rather be embedded in text lines, like this:

```
\*[COND]Not everything is as it seems.\*[CONDX]
```

***[COND]** remains in effect until you turn it off with ***[CONDX]**.

IMPORTANT: You must turn ***[COND]** off before making any changes to the point size of your type, either via the **.PT_SIZE** macro or with the **\s** inline escape sequence. If you wish the new point size to be pseudo-condensed, simply reinvoke ***[COND]** afterward. Equally, ***[COND]** must be turned off before changing the condense percentage with **.CONDENSE**.

Note: If you're using the document processing macros with **.PRINTSTYLE TYPEWRITE**, *mom* ignores ***[COND]** requests.

[CONDSUP]**...[CONDSUPX]**
pseudo-condensed superscript

[DOWN** *n]**
temporarily move downward in a line

***[EN–MARK]**
mark initial line of a range of line numbers (for use with line numbered endnotes)

***[EXT]**
***[EXTX]**
Pseudo-extending on/off

***[EXT]** begins pseudo-extending type. ***[EXTX]** turns the feature off. Both are inline escape sequences; therefore, they should not appear as separate lines, but rather be embedded in text lines, like this:

```
\*[EXT]Not everything is as it seems.\*[EXTX]
```

***[EXT]** remains in effect until you turn it off with ***[EXTX]**.

IMPORTANT: You must turn ***[EXT]** off before making any changes to the point size of your type, either via the **.PT_SIZE** macro or with the **\s** inline escape sequence. If you wish the new point size to be *pseudo-extended*, simply reinvoke ***[EXT]** afterward. Equally, ***[EXT]** must be turned off before changing the extend percentage with **.EXTEND**.

Note: If you are using the document processing macros with **.PRINTSTYLE TYPEWRITE**, *mom* ignores ***[EXT]** requests.

[EXTSUP]**...[EXTSUPX]**
pseudo extended superscript

[FU** *n]**
move characters pairs further apart inline (related to macro **.KERN**)

[FWD** *n]**
move forward in a line

***[LEADER]**
insert leaders at the end of a line

***[RULE]**
draw a full measure rule

[SIZE** *n]**
change the point size inline (related to macro **.PT_SIZE**)

***[SLANT]**
***[SLANTX]**
Pseudo italic on/off

***[SLANT]** begins *pseudo-italicizing type*. ***[SLANTX]** turns the feature off. Both are inline escape sequences; therefore, they should not appear as separate lines, but rather be embedded in text lines, like this:

```
Not \*[SLANT]everything\*[SLANTX] is as it seems.
```

Alternatively, if you wanted the whole line *pseudo-italicized*, you'd do

```
\*[SLANT]Not everything is as it seems.\*[SLANTX]
```

Once ***[SLANT]** is invoked, it remains in effect until turned off.

Note: If you're using the document processing macros with **.PRINTSTYLE TYPEWRITE**, *mom* underlines pseudo-italics by default. To change this behaviour, use the special macro **.SLANT_MEANS_SLANT**.

[ST<number***>]**...***[ST<***number***>X]**
Mark positions of string tabs

The *quad* direction must be **LEFT** or **JUSTIFY** (see **.QUAD** and **.JUSTIFY**) or the *no-fill mode* set to **LEFT** in order for these inlines to function properly. Please see *IMPORTANT*, below.

String tabs need to be marked off with inline escape sequences before being set up with the **.ST** macro. Any input line may contain string tab markers. *<number>*, above, means the numeric identifier of the tab.

The following shows a sample input line with string tab markers.

```
\*[ST1]De minimus\*[ST1X]non curat\*[ST2]lex\*[ST2X].
```

String *tab 1* begins at the start of the line and ends after the word *time*. String *tab 2* starts at *good* and ends after *men*. *Inline escape sequences* (e.g., *font* or *point size changes*, or horizontal movements, including padding) are taken into account when *mom* determines the *position* and *length* of *string tabs*.

Up to nineteen string tabs may be marked (not necessarily all on the same line, of course), and they must be numbered between 1 and 19.

Once string tabs have been marked in input lines, they have to be *set* with **.ST**, after which they may be called, by number, with **.TAB**.

Note: Lines with string tabs marked off in them are normal input lines, i.e. they get printed, just like any input line. If you want to set up string tabs without the line printing, use the **.SILENT** macro.

IMPORTANT: Owing to the way *groff* processes input lines and turns them into output lines, it is not possible for *mom* to *guess* the correct starting position of string tabs marked off in lines that are centered or set flush right.

Equally, she cannot guess the starting position if a line is fully justified and broken with **.SPREAD**.

In other words, in order to use string tabs, **LEFT** must be active, or, if **.QUAD LEFT** or **JUSTIFY** are active, the line on which the *string tabs* are marked must be broken *manually* with **.BR** (but not **.SPREAD**).

To circumvent this behaviour, I recommend using the **PAD** to set up string tabs in centered or flush right lines. Say, for example, you want to use a *string tab* to *underscore* the text of a centered line with a rule. Rather than this,

```
.CENTER
\*[ST1]A line of text\*[ST1X]\c
```

```
.EL
.ST 1
.TAB 1
.PT_SIZE 24
.ALD 3p
\*[RULE]
.RLD 3p
.TQ
```

you should do:

```
.QUAD CENTER
.PAD "#\*[ST1]A line of text\*[ST1X]#"
.EL
.ST 1
.TAB 1
.PT_SIZE 24
.ALD 3p
\" You can't use \*[UP] or \*[DOWN] with \*[RULE].
.RLD 3p
.TQ
```

[SUP]**...[SUPX]**
superscript

***[TB+]**
Inline escape for **.TN** (*Tab Next*)

[UL]**...[ULX]**
invoke underlining inline (fixed width fonts only)

[UP** *n]**
temporarily move upward in a line

Details of macros in alphabetical order

.AUTOLEAD
set the linespacing relative to the point size

.B_MARGIN <*bottom margin*>
Bottom Margin

Requires a unit of measure

.B_MARGIN sets a nominal position at the bottom of the page beyond which you don't want your type to go. When the bottom margin is reached, *mom* starts a new page. **.B_MARGIN requires a unit of measure.** Decimal fractions are allowed. To set a nominal bottom margin of 3/4 inch, enter

```
.B_MARGIN .75i
```

Obviously, if you haven't spaced the type on your pages so that the last lines fall perfectly at the bottom margin, the margin will vary from page to page. Usually, but not always, the last line of type that fits on a page before the bottom margin causes mom to start a new page.

Occasionally, owing to a peculiarity in *groff*, an extra line will fall below the nominal bottom margin. If you're using the document processing macros, this is unlikely to happen; the document processing macros are very hard-nosed about aligning bottom margins.

Note: The meaning of **.B_MARGIN** is slightly different when you're using the document processing macros.

.FALLBACK_FONT <*fallback font*> **[ABORT | WARN]**
Fallback Font

In the event that you pass an invalid argument to **.FAMILY** (i.e. a non-existent *family*), *mom*, by

default, uses the *fallback font*, **Courier Medium Roman** (**CR**), in order to continue processing your file.

If you'd prefer another *fallback font*, pass **.FALLBACK_FONT** the full *family+font name* of the *font* you'd like. For example, if you'd rather the *fallback font* were **Times Roman Medium Roman**,

```
.FALLBACK_FONT TR
```

would do the trick.

Mom issues a warning whenever a *font style set* with **.FT** does not exist, either because you haven't registered the style or because the *font style* does not exist in the current *family set* with **.FAMILY**. By default, **mom** then aborts, which allows you to correct the problem.

If you'd prefer that **mom** not abort on non-existent *fonts*, but rather continue processing using a *fallback font*, you can pass **.FALLBACK_FONT** the argument **WARN**, either by itself, or in conjunction with your chosen *fallback font***.**

Some examples of invoking **.FALLBACK_FONT**:

.FALLBACK_FONT WARN
: *mom* will issue a warning whenever you try to access a non-existent *font* but will continue processing your file with the default *fallback font*, **Courier Medium Roman**.

.FALLBACK_FONT TR WARN
: **mom** will issue a warning whenever you try to access a non-existent *font* but will continue processing your file with a *fallback font* of **Times Roman Medium Roman**; additionally, **TR** will be the *fallback font* whenever you try to access a *family* that does not exist.

.FALLBACK_FONT TR ABORT
: **mom** will abort whenever you try to access a non-existent **font**, and will use the *fallback font* **TR** whenever you try to access a *family* that does not exist. If, for some reason, you want to revert to **ABORT**, just enter **".FALLBACK_FONT ABORT"** and *mom* will once again abort on *font errors*.

.FAM *<family>*
: Type Family, alias of **.FAMILY**

.FAMILY *<family>*
: Type Family, alias of **.FAM**

.FAMILY takes one argument: the name of the *family* you want. *Groff* comes with a small set of basic families, each identified by a 1-, 2- or 3-letter mnemonic. The standard families are:

```
A   = Avant Garde
BM  = Bookman
H   = Helvetica
HN  = Helvetica Narrow
N   = New Century Schoolbook
P   = Palatino
T   = Times Roman
ZCM = Zapf Chancery
```

The argument you pass to **.FAMILY** is the identifier at left, above. For example, if you want **Helvetica**, enter

```
.FAMILY H
```

Note: The font macro (**.FT**) lets you specify both the type *family* and the desired font with a single macro. While this saves a few keystrokes, I recommend using **.FAMILY for** *family*, and **.FT for** *font*, except where doing so is genuinely inconvenient. **ZCM**, for example, only exists in one style: **Italic** (**I**).

Therefore,

```
.FT ZCMI
```

makes more sense than setting the *family* to **ZCM**, then setting the *font* to *I*.

Additional note: If you are running a *groff* version prior to 1.19.2, you must follow all **.FAMILY** requests with a **.FT** request, otherwise *mom* will set all type up to the next **.FT** request in the fallback font.

If you are running *groff* 1.19.2 or later, when you invoke the **.FAMILY** macro, *mom remembers* the font style (Roman**, Italic**, etc) currently in use (if the font style exists in the new *family*) and will continue to use the same font style in the new family. For example:

```
.FAMILY BM \" Bookman family
.FT I \" Medium Italic
<some text> \" Bookman Medium Italic
.FAMILY H \" Helvetica family
<more text> \" Helvetica Medium Italic
```

However, if the font style does not exist in the new family, *mom* will set all subsequent type in the fallback font (by default, **Courier Medium Roman**) until she encounters a **.FT** request that's valid for the *family*.

For example, assuming you don't have the font **Medium Condensed Roman** (*mom* extension *CD*) in the *Helvetica family*:

```
.FAMILY UN \" Univers family
.FT CD \" Medium Condensed
<some text> \" Univers Medium Condensed
.FAMILY H \" Helvetica family
<more text> \" Courier Medium Roman!
```

In the above example, you must follow **.FAMILY H** with a **.FT** request that's valid for **Helvetica**.

Please see the Appendices, *Adding fonts to groff*, for information on adding fonts and families to *groff*,*as*well*as*to see a list of the extensions *mom* provides to *groff*'s basic **R**, **I**, **B**, **BI** styles.

Suggestion: When adding *families to groff*, I recommend following the established standard for the naming families and fonts. For example, if you add the **Garamond** family, name the font files

```
GARAMONDR
GARAMONDI
GARAMONDB
GARAMONDBI
```

GARAMOND then becomes a valid *family name* you can pass to **.FAMILY**. (You could, of course, shorten **GARAMOND** to just **G**, or **GD**.) **R**, **I**, **B**, and **BI** after **GARAMOND** are the *roman*, *italic*, *bold* and *bold-italic* fonts respectively.

.FONT R | B | BI | *<any other valid font style>*
Alias to **.FT**

.FT R | B | BI | *<any other valid font style>*
Set font

By default, *groff* permits **.FT** to take one of four possible arguments specifying the desired font:

```
R = (Medium) Roman
I = (Medium) Italic
B = Bold (Roman)
BI = Bold Italic
```

For example, if your *family* is **Helvetica**, entering

```
.FT B
```

will give you the *Helvetica bold font*. If your *family* were **Palatino**, you'd get the *Palatino bold font*.

Mom considerably extends the range of arguments you can pass to **.FT**, making it more convenient to add and access fonts of differing weights and shapes within the same family.

Have a look here for a list of the weight/style arguments *mom* allows. Be aware, though, that you must have the fonts, correctly installed and named, in order to use the arguments. (See *Adding fonts to groff* for instructions and information.) Please also read the *ADDITIONAL NOTE* found in the description of the **.FAMILY** macro.

How *mom* reacts to an invalid argument to **.FT** depends on which version of *groff* you're using. If your *groff* version is 1.19.2 or later, *mom* will issue a warning and, depending on how you've set up the fallback font, either continue processing using the fallback font, or abort (allowing you to correct the problem). In earlier versions, *mom* will silently continue processing, using either the fallback font or the font that was in effect prior to the invalid **.FT** call.

.FT will also accept, as an argument, a full *family* and *font name*.

For example,

```
.FT HB
```

will set subsequent type in *Helvetica Bold*.

However, I strongly recommend keeping *family* and *font* separate except where doing so is genuinely inconvenient.

For inline control of *fonts*, see *Inline Escapes*, font control.

.HI [<*measure*> **]**

Hanging indent — the optional argument requires a unit of measure.

A hanging indent looks like this:

```
The thousand injuries of Fortunato I had borne as best I
  could, but when he ventured upon insult, I vowed
  revenge.  You who so well know the nature of my soul
  will not suppose, however, that I gave utterance to a
  threat, at length I would be avenged...
```

The first line of text *hangs* outside the *left margin*.

In order to use *hanging indents*, you must first have a *left indent* active (set with either **.IL** or **.IB**). **Mom** will not hang text outside the *left margin set* with **.L_MARGIN** or outside the *left margin* of a *tab*.

The first time you invoke **.HI**, you must give it a **measure**. If you want the first line of a paragraph to *hang by*, say, *1 pica*, do

```
.IL 1P
.HI 1P
```

Subsequent invocations of **.HI** do not require you to supply a *measure*; *mom* keeps track of the last measure you gave it.

Generally speaking, you should invoke **.HI** immediately prior to the line you want hung (i.e. without any intervening control lines). And because *hanging indents* affect only one line, there's no need to turn them off.

IMPORTANT: Unlike **IL**, **IR** and **IB**, measures given to **.HI** are NOT additive. Each time you pass a measure to **.HI**, the measure is treated literally. *Recipe:* A numbered list using *hanging indents*

Note: mom has macros for setting lists. This recipe exists to demonstrate the use of *hanging indents* only.

```
.PAGE 8.5i 11i 1i 1i 1i 1i
.FAMILY  T
.FT      R
.PT_SIZE 12
.LS      14
.JUSTIFY
.KERN
.SS 0
```

```
.IL \w'\0\0.'
.HI \w'\0\0.'
1.\0The most important point to be considered is whether
the answer to the meaning of Life, the Universe, and
Everything really is 42.  We have no one's word on the
subject except Mr. Adams's.
.HI
2.\0If the answer to the meaning of Life, the Universe,
and Everything is indeed 42, what impact does this have on
the politics of representation?  42 is, after all not a
prime number.  Are we to infer that prime numbers don't
deserve equal rights and equal access in the universe?
.HI
3.\0If 42 is deemed non-exclusionary, how do we present
it as the answer and, at the same time, forestall debate
on its exclusionary implications?
```

First, we invoke a left indent with a measure equal to the width of 2 figures spaces plus a period (using the \w inline escape). At this point, the left indent is active; text afterward would normally be indented. However, we invoke a hanging indent of exactly the same width, which hangs the first line (and first line only!) to the left of the indent by the same distance (in this case, that means "out to the left margin"). Because we begin the first line with a number, a period, and a figure space, the actual text (*The most important point...*) starts at exactly the same spot as the indented lines that follow.

Notice that subsequent invocations of **.HI** don't require a *measure* to be given.

Paste the example above into a file and preview it with

pdfmom filename.mom | ps2pdf - filename.pdf

to see hanging indents in action.

.IB [*<left measure>* *<right measure>* **]**

Indent both — the optional argument requires a unit of measure

.IB allows you to set or invoke a left and a right indent at the same time.

At its first invocation, you must supply a measure for both indents; at subsequent invocations when you wish to supply a measure, both must be given again. As with **.IL** and **.IR**, the measures are added to the values previously passed to the macro. Hence, if you wish to change just one of the values, you must give an argument of zero to the other.

A word of advice: If you need to manipulate left and right indents separately, use a combination of **.IL** and **.IR** instead of **.IB**. You'll save yourself a lot of grief.

A *minus sign* may be prepended to the arguments to subtract from their current values. The \w inline escape may be used to specify text-dependent measures, in which case no unit of measure is required. For example,

.IB \w'margarine' \w'jello'

left indents text by the width of the word *margarine* and right indents by the width of *jello*.

Like **.IL** and **.IR**, **.IB** with no argument indents by its last active values. See the brief explanation of how mom handles indents for more details.

Note: Calling a *tab* (with **.TAB <n>**) automatically cancels any active indents.

Additional note: Invoking **.IB** automatically turns off **.IL** and **.IR**.

.IL [*<measure>* **]**

Indent left — the optional argument requires a unit of measure

.IL indents text from the left margin of the page, or if you're in a *tab*, from the left edge of the *tab*. Once *IL* is on, the *left indent* is applied uniformly to every subsequent line of text, even if you

change the line length.

The first time you invoke **.IL**, you must give it a measure. Subsequent invocations with a measure add to the previous measure. A minus sign may be prepended to the argument to subtract from the current measure. The **\w** inline escape may be used to specify a text-dependent measure, in which case no unit of measure is required. For example,

```
.IL \w'margarine'
```

indents text by the width of the word *margarine*.

With no argument, **.IL** indents by its last active value. See the brief explanation of how *mom* handles indents for more details.

Note: Calling a *tab* (with **.TAB <n>**) automatically cancels any active indents.

Additional note: Invoking **.IL** automatically turns off **IB**.

.IQ [*<measure>* **]**

IQ — quit any/all indents

IMPORTANT NOTE: The original macro for quitting all indents was **.IX**. This usage has been deprecated in favour of **IQ**. **.IX** will continue to behave as before, but *mom* will issue a warning to *stderr* indicating that you should update your documents.

As a consequence of this change, **.ILX**, **.IRX** and **.IBX** may now also be invoked as **.ILQ**, **.IRQ** and **.IBQ**. Both forms are acceptable.

Without an argument, the macros to quit indents merely restore your original margins and line length. The measures stored in the indent macros themselves are saved so you can call them again without having to supply a measure.

If you pass these macros the optional argument **CLEAR**, they not only restore your original left margin and line length, but also clear any values associated with a particular indent style. The next time you need an indent of the same style, you have to supply a measure again.

.IQ CLEAR, as you'd suspect, quits and clears the values for all indent styles at once.

.IR [*<measure>* **]**

Indent right — the optional argument requires a unit of measure

.IR indents text from the *right margin* of the page, or if you're in a *tab*, from the end of the *tab*.

The first time you invoke **.IR**, you must give it a measure. Subsequent invocations with a measure add to the previous indent measure. A *minus sign* may be prepended to the argument to subtract from the current indent measure. The \w inline escape may be used to specify a text-dependent measure, in which case no *unit of measure* is required. For example,

```
.IR \w'jello'
```

indents text by the width of the word *jello*.

With no argument, **.IR** indents by its last active value. See the brief explanation of how *mom* handles indents for more details.

Note: Calling a *tab* (with **.TAB <n>**) automatically cancels any active indents.

Additional note: Invoking **.IR** automatically turns off **IB**.

.L_MARGIN *<left margin>*

Left Margin

L_MARGIN establishes the distance from the left edge of the printer sheet at which you want your type to start. It may be used any time, and remains in effect until you enter a new value.

Left indents and tabs are calculated from the value you pass to **.L_MARGIN**, hence it's always a good idea to invoke it before starting any serious typesetting. A unit of measure is required. Decimal fractions are allowed. Therefore, to set the left margin at 3 picas (1/2 inch), you'd enter either

```
.L_MARGIN 3P
```

or

```
.L_MARGIN .5i
```

If you use the macros **.PAGE**, **.PAGEWIDTH** or **.PAPER** without invoking **.L_MARGIN** (either before or afterward), *mom* automatically sets **.L_MARGIN** to *1 inch*.

Note: **.L_MARGIN** behaves in a special way when you're using the document processing macros.

.MCO Begin multi-column setting.

.MCO (*Multi-Column On*) is the *macro* you use to begin *multi-column setting*. It marks the current baseline as the top of your columns, for use later with **.MCR**. See the introduction to columns for an explanation of *multi-columns* and some sample input.

Note: Do not confuse **.MCO** with the **.COLUMNS** macro in the document processing macros.

.MCR Once you've turned *multi-columns* on (with **.MCO**), **.MCR**, at any time, returns you to the *top of your columns*.

.MCX [<*distance to advance below longest column*> **]**
Optional argument requires a unit of measure.

Exit multi-columns.

.MCX takes you out of any *tab* you were in (by silently invoking **.TQ**) and advances to the bottom of the longest column.

Without an argument, **.MCX** advances *1 linespace* below the longest column.

Linespace, in this instance, is the leading in effect at the moment **.MCX** is invoked.

If you pass the <*distance*> argument to **.MCX**, it advances *1 linespace* below the longest column (see above) *PLUS* the distance specified by the argument. The argument requires a unit of measure; therefore, to advance an extra 6 points below where **.MCX** would normally place you, you'd enter

```
.MCX 6p
```

Note: If you wish to advance a precise distance below the baseline of the longest column, use **.MCX** with an argument of **0** (zero; no *unit of measure* required) in conjunction with the **.ALD** macro, like this:

```
.MCX 0
.ALD 24p
```

The above advances to precisely *24 points* below the baseline of the longest column.

.NEWPAGE

Whenever you want to start a new page, use **.NEWPAGE**, by itself with no argument. **Mom** will finish up processing the current page and move you to the top of a new one (subject to the top margin set with **.T_MARGIN**).

.PAGE <*width*> **[** <*length*> **[** <*lm*> **[** <*rm*> **[** <*tm*> **[** <*bm*> **]]]]]**

All arguments require a unit of measure

IMPORTANT: If you're using the document processing macros, **.PAGE** must come after **.START**. Otherwise, it should go at the top of a document, prior to any text. And remember, when you're using the document processing macros, top margin and bottom margin mean something slightly different than when you're using just the typesetting macros (see Top and bottom margins in document processing).

.PAGE lets you establish paper dimensions and page margins with a single macro. The only required argument is page width. The rest are optional, but they must appear in order and you can't skip over any. <*lm*>, <*rm*>, <*tm*> and <*bm*> refer to the left, right, top and bottom margins respectively.

Assuming your page dimensions are 11 inches by 17 inches, and that's all you want to set, enter

```
.PAGE 11i 17i
```

If you want to set the left margin as well, say, at 1 inch, **PAGE** would look like this:

```
.PAGE 11i 17i 1i
```

Now suppose you also want to set the top margin, say, at 1–1/2 inches. *<tm>* comes after *<rm>* in the optional arguments, but you can't skip over any arguments, therefore to set the top margin, you must also give a right margin. The **.PAGE** macro would look like this:

```
.PAGE 11i 17i 1i 1i 1.5i
                    |   |
required right---+  +---top margin
        margin
```

Clearly, **.PAGE** is best used when you want a convenient way to tell *mom* just the dimensions of your printer sheet (width and length), or when you want to tell her everything about the page (dimensions and all the margins), for example

```
.PAGE 8.5i 11i 45p 45p 45p 45p
```

This sets up an 8½ by 11 inch page with margins of 45 points (5/8-inch) all around.

Additionally, if you invoke **.PAGE** with a top margin argument, any macros you invoke after **.PAGE** will almost certainly move the baseline of the first line of text down by one linespace. To compensate, do

```
.RLD 1v
```

immediately before entering any text, or, if it's feasible, make **.PAGE** the last macro you invoke prior to entering text.

Please read the *Important note* on page dimensions and papersize for information on ensuring *groff* respects your **.PAGE** dimensions and margins.

.PAGELENGTH *<length of printer sheet>*

tells *mom* how long your printer sheet is. It works just like **.PAGEWIDTH**.

Therefore, to tell *mom* your printer sheet is 11 inches long, you enter

```
.PAGELENGTH 11i
```

Please read the important note on page dimensions and papersize for information on ensuring *groff* respects your *PAGELENGTH*.

.PAGEWIDTH *<width of printer sheet>*

The argument to **.PAGEWIDTH** is the width of your printer sheet.

.PAGEWIDTH requires a unit of measure. Decimal fractions are allowed. Hence, to tell *mom* that the width of your printer sheet is 8½ inches, you enter

```
.PAGEWIDTH 8.5i
```

Please read the Important note on page dimensions and papersize for information on ensuring *groff* respects your *PAGEWIDTH*.

.PAPER *<paper type>*

provides a convenient way to set the page dimensions for some common printer sheet sizes. The argument *<paper type>* can be one of: **LETTER**, **LEGAL**, **STATEMENT**, **TABLOID**, **LEDGER**, **FOLIO**, **QUARTO**, **EXECUTIVE**, **10x14**, **A3**, **A4**, **A5**, **B4**, **B5**.

.PRINTSTYLE

.PT_SIZE *<size of type in points>*

Point size of type, does not require a *unit of measure*.

.PT_SIZE (*Point Size*) takes one argument: the *size of type* in *points*. Unlike most other macros that establish the *size* or *measure* of something, **.PT_SIZE** does not require that you supply a *unit of measure* since it's a near universal convention that *type size* is measured in *points*. Therefore, to change the *type size* to, say, *11 points*, enter

```
.PT_SIZE 11
```

Point sizes may be *fractional* (e.g., *10.25* or *12.5*).

You can prepend a *plus* or a *minus sign* to the argument to **.PT_SIZE**, in which case the *point size* will be changed by + or − the original value. For example, if the *point size* is *12*, and you want *14*, you can do

```
.PT_SIZE +2
```

then later reset it to *12* with

```
.PT_SIZE -2
```

The *size of type* can also be changed inline.

Note: It is unfortunate that the **pic** preprocessor has already taken the name, PS, and thus *mom*'s macro for setting *point sizes* can't use it. However, if you aren't using **pic**, you might want to alias **.PT_SIZE** as **.PS**, since there'd be no conflict. For example

```
.ALIAS PS PT_SIZE
```

would allow you to set *point sizes* with **.PS**.

.R_MARGIN *<right margin>*

Right Margin

Requires a unit of measure.

IMPORTANT: **.R_MARGIN**, if used, must come after **.PAPER**, **.PAGEWIDTH**, **.L_MARGIN**, and/or **.PAGE** (if a right margin isn't given to PAGE). The reason is that **.R_MARGIN** calculates line length from the overall page dimensions and the left margin.

Obviously, it can't make the calculation if it doesn't know the page width and the left margin.

.R_MARGIN establishes the amount of space you want between the end of typeset lines and the right hand edge of the printer sheet. In other words, it sets the line length. **.R_MARGIN** requires a unit of measure. Decimal fractions are allowed.

The line length macro (LL) can be used in place of **.R_MARGIN**. In either case, the last one invoked sets the line length. The choice of which to use is up to you. In some instances, you may find it easier to think of a section of type as having a right margin. In others, giving a line length may make more sense.

For example, if you're setting a page of type you know should have 6-pica margins left and right, it makes sense to enter a left and right margin, like this:

```
.L_MARGIN 6P
.R_MARGIN 6P
```

That way, you don't have to worry about calculating the line length. On the other hand, if you know the line length for a patch of type should be 17 picas and 3 points, entering the line length with LL is much easier than calculating the right margin, e.g.,

```
.LL 17P+3p
```

If you use the macros **.PAGE**, **.PAGEWIDTH** or **PAPER** without invoking **.R_MARGIN** afterward, *mom* automatically sets **.R_MARGIN** to *1 inch*. If you set a line length after these macros (with **.LL**), the line length calculated by **.R_MARGIN** is, of course, overridden.

Note: **.R_MARGIN** behaves in a special way when you're using the document processing macros.

.ST *<tab number>* **L | R | C | J [QUAD]**

After *string tabs* have been marked off on an input line (see ***[ST]...*[STX]**), you need to *set* them by giving them a direction and, optionally, the **QUAD** argument.

In this respect, **.ST** is like **.TAB_SET** except that you don't have to give **.ST** an indent or a line length (that's already taken care of, inline, by ***[ST]...*[STX]**).

If you want string *tab 1* to be **left**, enter

```
.ST 1 L
```

If you want it to be *left* and *filled*, enter

```
.ST 1 L QUAD
```

If you want it to be justified, enter

```
.ST 1 J
```

.TAB *<tab number>*

After *tabs* have been defined (either with **.TAB_SET** or **.ST**), **.TAB** moves to whatever *tab number* you pass it as an argument.

For example,

```
.TAB 3
```

moves you to *tab 3*.

Note: **.TAB** breaks the line preceding it and advances 1 linespace. Hence,

```
.TAB 1
A line of text in tab 1.
.TAB 2
A line of text in tab 2.
```

produces, on output

```
A line of text in tab 1.
                                    A line of text in tab 2.
```

If you want the tabs to line up, use **.TN** ("Tab Next") or, more conveniently, the inline escape sequence ***[TB+]**:

```
.TAB 1
A line of text in tab 1.\*[TB+]
A line of text in tab 2.
```

which produces

```
A line of text in tab 1.   A line of text in tab 2.
```

If the text in your tabs runs to several lines, and you want the first lines of each tab to align, you must use the multi-column macros.

Additional note: Any indents in effect prior to calling a tab are automatically turned off by **TAB**. If you were happily zipping down the page with a left indent of *2 picas* turned on, and you call a *tab* whose indent from the left margin is *6 picas*, your new distance from the *left margin* will be *6 picas*, not I 6 picas plus the 2 pica indent.

Tabs are not by nature columnar, which is to say that if the text inside a *tab* runs to several lines, calling another *tab* does not automatically move to the baseline of the first line in the *previous tab*. To demonstrate:

```
TAB 1
Carrots
Potatoes
Broccoli
.TAB 2
$1.99/5 lbs
$0.25/lb
$0.99/bunch
```

produces, on output

```
Carrots
Potatoes
Broccoli
            $1.99/5 lbs
            $0.25/lb
            $0.99/bunch
```

.TB *<tab number>*

Alias to **.TAB**

.TI [*<measure>* **]**

Temporary left indent — the optional argument requires a *unit of measure*

A temporary indent is one that applies only to the first line of text that comes after it. Its chief use is indenting the first line of paragraphs. (**Mom's .PP** macro, for example, uses a *temporary indent*.)

The first time you invoke **.TI**, you must give it a measure. If you want to *indent* the first line of a paragraph by, say, 2 ems, do

```
.TI 2m
```

Subsequent invocations of **.TI** do not require you to supply a measure; *mom* keeps track of the last measure you gave it.

Because *temporary indents* are temporary, there's no need to turn them off.

IMPORTANT: Unlike **.IL**, **.IR** and **IB**, measures given to **.TI** are NOT additive. In the following example, the second **".TI 2P"** is exactly *2 picas*.

```
.TI 1P
The beginning of a paragraph. . .
.TI 2P
The beginning of another paragraph. . .
```

.TN Tab Next

Inline escape ***[TB+]**

TN moves over to the *next tab* in numeric sequence (*tab n+1*) without advancing on the page. See the *NOTE* in the description of the **.TAB** macro for an example of how **TN** works.

In *tabs* that aren't given the **QUAD** argument when they're set up with **.TAB_SET** or **ST**, you must terminate the line preceding **.TN** with the **\c** inline escape sequence. Conversely, if you did give a **QUAD** argument to **.TAB_SET** or **ST**, the **\c must not be used.**

If you find remembering whether to put in the **\c** bothersome, you may prefer to use the inline escape alternative to **.TN**, ***[TB+]**, which works consistently regardless of the fill mode.

Note: You must put text in the input line immediately after **.TN**. Stacking of **.TN**'s is not allowed. In other words, you cannot do

```
.TAB 1
Some text\c
.TN
Some more text\c
.TN
.TN
Yet more text
```

The above example, assuming *tabs* numbered from *1* to *4*, should be entered

```
.TAB 1
Some text\c
.TN
Some more text\c
.TN
\&\c
.TN
Yet more text
```

\& is a zero-width, non-printing character that *groff* recognizes as valid input, hence meets the requirement for input text following **.TN**.

.TQ **TQ** takes you out of whatever *tab* you were in, advances *1 linespace*, and restores the *left margin*, *line length*, *quad direction* and *fill mode* that were in effect prior to invoking any *tabs*.

.T_MARGIN *<top margin>*

Top margin

Requires a unit of measure

.T_MARGIN establishes the distance from the top of the printer sheet at which you want your type to start. It requires a unit of measure, and decimal fractions are allowed. To set a top margin of 2½ centimetres, you'd enter

```
.T_MARGIN 2.5c
```

.T_MARGIN calculates the vertical position of the first line of type on a page by treating the top edge of the printer sheet as a baseline. Therefore,

```
.T_MARGIN 1.5i
```

puts the baseline of the first line of type 1½ inches beneath the top of the page.

Note: **.T_MARGIN** means something slightly different when you're using the document processing macros. See Top and bottom margins in document processing for an explanation.

IMPORTANT: **.T_MARGIN** does two things: it establishes the top margin for pages that come after it and it moves to that position on the current page. Therefore, **.T_MARGIN** should only be used at the top of a file (prior to entering text) or after NEWPAGE, like this:

```
.NEWPAGE
.T_MARGIN 6P
<text>
```

Authors

mom was written by Peter Schaffter ⟨peter@schaffter.ca⟩. PDF support was provided by Deri James ⟨deri@chuzzlewit.myzen.co.uk⟩. This manual page was written by Bernd Warken.

See also

/usr/local/share/doc/groff−1.23.0/html/mom/toc.html
: entry point to the HTML documentation

⟨http://www.schaffter.ca/mom/momdoc/toc.html⟩
: HTML documentation online

⟨http://www.schaffter.ca/mom/⟩
: the *mom* macros homepage

Groff: The GNU Implementation of troff, by Trent A. Fisher and Werner Lemberg, is the primary *groff* manual. You can browse it interactively with "info groff".

pdfmom(1), *groff*(1), *troff*(1)