Groff and mom: an overview

by

Peter Schaffter

CONTENTS

Groff and mom: an overview

by

Peter Schaffter

Not everyone coming to *mom* for the first time is familiar with groff, the program responsible for typesetting files formatted with *mom*. Equally, those already acquainted with groff from traditional macro packages such as *ms* or *me* may find some aspects of her approach to formatting unusual. This document aims to be of use to both user groups by outlining basic groff concepts while exploring key components of mom.

Groff

In broad terms, the groff document formatting system comprises two parts: a typesetter and a formatting language. The language permits 'if-else' and 'while' constructs, as well as variable and string manipulation, making it a specialized, albeit rudimentary, programming language. Its main purpose is for writing macros, which group low-level typesetting requests into meaningful instructions, for example '**.PP**' to begin a paragraph, or '**.UNDERSCORE**' to place a rule beneath text.[1]

▪ Workflow

Documents to be typeset with groff begin as plain text files ("source files") interspersed with formatting instructions. The instructions may be either **macros** or **requests**, and appear on a line by themselves, beginning with a period. **Escape sequences** may also be used to perform typographic manipulation.

When a source file is ready to be typeset, it is passed to groff at the command line. Text is justified, or not, and requests and macros executed as they are encountered. The resulting output, redirected to a file, is ready to be viewed or printed.

▪ Input processing

Effective use of groff benefits from understanding how input files are processed.

Lines from source files are read and processed in the order they appear. If a line begins with a period, it is treated as a request or macro and executed immediately. All other lines are considered formattable text.

If the text is being justified, groff collects words from the current and subsequent input lines until the desired output line length is reached, at which point the line is justified and groff begins collecting words again. If the text is not being justified, each input line corresponds to a single output line.

This line-at-a-time approach entails limitations. Some typesetting decisions have to be made in advance of groff having the whole picture. Groff cannot, for example,

skip to the end of a paragraph to determine optimal line breaks, or scan an entire page to equalize whitespace. The user is expected to effect pleasing solutions to these and other common typographical issues with macros and requests.

The approach also imposes a certain structure on source files, since, unavoidably, some requests and macros must precede others.

▪ Requests

Requests are the low-level typesetting commands that form the heart of groff. Typically terse, often just two or three lowercase letters, they are exemplars of the Unix philosophy: *Small tools that do one job, and do it well.* They are also notoriously difficult to fathom. Their behaviour and interaction can seem whimsical at times, despite being fully and accurately documented.[2]

It is unusual for a document to be formatted with groff requests alone. Users are expected to format with macros from a **macro package**. Nevertheless, familiarity with groff requests aids in coping with typographic challenges.

▪ Escape sequences

In addition to requests, groff permits inline formatting* through the use of escape sequences. Introduced by the backslash character, escape sequences perform a variety of functions, from changing fonts and point sizes to performing drawing operations such as rules and boxes.

As with requests, knowledge of escape sequences is an asset.

▪ Number registers

While not strictly analogous, number registers are essentially variables, accessed with the construct **`\n[reg]`**. Groff's built-in registers hold complete information about the state of a document at any given point: type parameters, justification style, vertical position, and so on.

Mostly used in writing macros, number registers have their place in source files. A rule, for example, can be drawn to half the current line length using **`\n[.l]/2`** as the length argument to '**`.DRH`**', where **`\n[.l]`** is the built-in register holding the current line length.[3]

When formatting with one of the **classical macro packages**, number registers are also used to set flags and pass arguments to macros. Users must set some registers themselves with the '**`.nr`**' request. For example, when using the *ms* macros, '**`.nr LL 5i`**' sets the default line length to five inches.

▪ String registers

String registers are similar to number registers except they hold text instead of numeric values. Created with' **`.ds string-name string`**' and accessed with the

*Typesetting instructions embedded in the body of the text.

construct ***[string-name]**, they are useful in source files to save repeated typing, and serve to set flags and hold user-supplied input in macros.

▪ Macros and macro packages

Most formatting for groff is done with macros, which group common, routine operations into convenient single commands. As such, they form the primary user interface to groff. A collection of macros is referred to as a "macro package" or "macro set." By convention, the macros have uppercase names to distinguish them from groff requests.

Nearly all groff documents are formatted with one of the classical macro packages—*ms*, *mm*, *me*—or *mom*, which must be named explicitly on the command line.[4] All provide similar facilities for basic formatting: page headers and footers, headings, paragraphs, footnotes, etc. Additional features and conveniences determine the choice of package.

To a greater or lesser extent, macro packages impose a certain style on documents. The classical macros exhibit a bias toward technical reports and papers, while *mom* leans toward the humanities and places greater emphasis on typographic refinements. Users can, however, adapt any package to their needs with low-level requests and supplementary macros.

▪ Preprocessors

Preprocessors have been written to simplify a number of complex typesetting tasks: **tbl** (tables), **eqn** (equations), **pic** (diagrams), and others. The**refer** preprocessor formats references and works cited from a bibliographic database.

In source files, preprocessor data and commands are entered between a pair of macros reserved for use by the preprocessor, which, moreover, must be called explicitly on the command line.[5]

▪ Output

Groff outputs PostScript or PDF, as well as formatted copy suitable for viewing at the terminal.[6] The output driver ("device") must called at the command line unless groff's default PostScript output is desired. Both PostScript and PDF output must be redirected to a file.

Mom

Mom overcomes two issues that have historically discouraged widespread adoption of groff: the classical macro packages' bias toward technical papers and reports, and the level of groff proficiency expected from the user. The latter is dealt with by insulating the user, inasmuch as possible, from the need for low-level formatting, while the former is addressed by providing flexible control over the style of a document.

▪ Two categories of macros

The macros in *mom* are grouped into two categories: typesetting macros and document processing macros.

The typesetting macros assist with *presentational* markup and include basic operations such as setting line lengths, establishing family and font, changing point size, and altering leading.* Additionally, they simplify a number of common typesetting routines and assist with typographic refinements.

The document processing macros cover *semantic* markup ('**.PP**' to start a paragraph, '**.FOOTNOTE**' to insert footnotes, etc) and the tools for styling all the semantic elements found in a document.

THE TYPESETTING MACROS

The typesetting macros are for typesetting directly, ie with no semantic markup.† Some have counterparts in groff requests; others are unique to *mom*. They can be used in conjunction with the document processing macros for occasional presentational markup, or by themselves.[7] Beyond the basic operations, they handle

– rag and justification styles
– inline character-pair kerning
– line ("track") kerning
– word and sentence spacing
– hyphenation policy
– vertical and horizontal spacing
– indenting (left, right, both, temporary, hanging)
– primitive graphical objects (rules, boxes, ellipses)
– coloured text
– artificial fonts (pseudo-italic, -bold, -condensed, -extended)

and perform other typesetting refinements. *Mom* also introduces some concepts not found in other macro packages:

- **autoleading** – updates the leading from an absolute value or scaling factor whenever the point size is changed with '**.PT_SIZE**';[8] use of autolead in conjunction with the document processing macros should be reserved for occasional presentational markup, and is disabled whenever a **document element tag** is called;[9]
- **padding** – takes whitespace remaining at the end of a line and inserts it at specified points in the line, in user-specified increments;
- **typesetting tabs**– tabs with a unique numeric identifier; each tab is the concatenation of an indent, a measure, a quad direction, and a justification style;

Leading* is used throughout this document in preference to the term "line spacing" to avoid confusion with groff's '.ls**' request, which sets line spacing policy (e.g. single-spaced vs double-spaced).

†Not all documents need titles, headings, and other sematinc markup—order forms, for example, or a wedding invitation.

- **string tabs** – tabs with a unique numeric identifier, created by specifying start and end points in a line of text; in conjunction with padding, simplifies the creation of scalable tab structures derived from text-widths.

THE DOCUMENT PROCESSING MACROS

Document processing can be summarized as the act of laying out pages of running text with consistent presentational features that reflect a document's logical structure and flow.

A well-formed *mom* source file has a basic structure that starts with metadata: author, title, chapter, and so on. Next comes a stylesheet section, which is introduced by the macro '**.PRINTSTYLE**'. The stylesheet may be sourced from a file.

After the stylesheet, the macro '**.START**' initiates document processing proper, and prepares mom for the body of the text. Throughout the text, sensibly-named macros identify the document's semantic ("logical") elements: '**.HEADING**' to identify nested levels of headings, '**.FOOTNOTE**' for footnotes, '**.QUOTE**' for cited material, etc.

Finally, if endnotes, a bibliography, or a table of contents is desired, the macros to output them come at the end of the source file.

Besides the document element tags, mom provides utilities for:

- **nested lists** – hierarchical list structures with user-specifiable enumerators;
- **line numbering** – a flexible system for prepending line numbers to text;
- **margin notes** – sensible handling of margin notes' vertical placement;
- **insertion of graphics** – in either PDF or PostScript format;*
- **floats** – reserved blocks of text, graphics, or preprocessor output that are kept together and deferred to the next page if insufficient room prevents their immediate output.

▪ Chapters as discrete documents

Mom treats each chapter of a document as a document unto itself. The end of each chapter in a multi-chapter document is signalled by '**.COLLATE**', and the beginning of the next by '**.START**'. Changes to metadata and style may come between the two.

If the chapters are saved as discrete files, they may be assembled by concatenating them and piping the output through groff.† Alternatively, the user may assemble a monolithic file from the outset.

▪ Printstyles

A unique feature of mom is the '**.PRINTSTYLE**' macro, which determines whether formatted copy is to be typeset or typewritten. The latter formats files correctly

---

*Other image formats must be converted; use of ImageMagick's **convert** utility is recommended.

†See **cat**(1).

for "typewritten/double-spaced" in a fixed-width font. Changing the argument to '**.PRINTSTYLE**' to '**TYPESET**' produces typeset copy instead, making it possible to generate both drafts and final output copy from the same source file.

- Document element tags

Macros that specifically identify semantic elements in a document are referred to as document element tags. They are either entered on a line by themselves (e.g. '**.PP**' to begin a paragraph), or require text as an argument on the same line ('**.HEADING <n> "text"**' for nested heading levels), or act as toggles enclosing a block of text ('**.EPIGRAPH**'/'**.EPIGRAPH off**').*

- Control macros

The styling of document element tags is managed by "control macros," which relieve users of manipulating number registers. For every tag, as is appropriate, there are control macros for family, font, size, leading, quad, indent, and colour. The naming scheme is natural language based, such that '**.PARA_INDENT**' establishes the first-line indent of paragraphs, while '**.FOOTNOTE_FAMILY**' sets the family for footnotes.

- Non-semantic elements

Non-semantic elements include page headers, footers, page numbering, and any other element that is not part of the running text. These, too, have control macros for setting typographic parameters.

- Page balancing

The insertion of headings, images, and other non-textual elements into a document, or spacing the paragraphs, can result in text that falls short of the bottom margin. Page balancing refers to how *mom* fills the page frame so that bottom margins remain equal ("flush").

THE PAGE FRAME

The page frame is the area of a page occupied by running text, bounded by white-space that forms the margins. The left and right margins are usually empty. The top and bottom margins typically contain document metadata (title, author) and page numbers; material contained within the top and bottom margins is referred to as "the header" or a "the footer."

Mom considers the distance from page top to the top of the frame as the top margin of a document, not the distance from page top to the header. Equally, the bottom margin is measured from the bottom of the page to the last line of running

*Any argument closes, or turns off, toggle macros. Users may choose their own convention, e.g. **off**, **QUIT**, **Close**, **X**, etc.

text, not from page bottom to the footer.* This allows the vertical placement of the headers and footers to be established independently of the page frame.

ADJUSTED LEADING

A document's requested leading rarely fills the page frame entirely to the bottom margin. Generally, the last line of text falls a few points short.† To compensate, *mom* introduces the notion of adjusted leading.‡ **Machine units** are added incrementally to the document's requested leading until the baseline grid fills the page frame and the last line of text falls precisely on the bottom margin. The difference between the requested leading and the adjusted leading is very slight, typically less than two hundred machine units, or 1/5 of a point.

If the user does not want adjusted leading, it may be disabled.

BASELINE GRID

Within the page frame, *mom* establishes a fixed grid of baselines from the leading requested before document processing begins.** The placement of all document element tags is relative to this grid. Whenever an element's vertical spacing requires placing it off the grid, compensation must applied—before, after, or both. *Mom* provides two strategies for compensating: shimming or flex-spacing.†† Except in impossible situations, e.g. insufficient room for a heading with at least one line of text underneath, these guarantee that the page frame is properly filled and that bottom margins are flush.

SHIMMING

When shimming is enabled, which it is by default, any element of running text that strays off the baseline grid is compensated for by *mom* nudging the line afterwards onto the next valid baseline. This ensures proper page fill and bottom margins that align from page to page.

Mom performs shimming automatically for a number of document element tags (headings, cited material, floats, etc). Users may also—indeed should—apply it themselves with the '**.SHIM**' macro after any disruption to the grid, for example a user-introduced change of leading on the page.

The default shimming of document tags occasionally results in the appearance of too much whitespace between elements, notably when the amount of shimming applied is close to the current leading value. An explicit negative vertical movement (e.g. '**.RLD 1v**' or '**.SP -1**') compensates‡‡

---

*Footnotes are considered part of the page frame.

†A PostScript point, the unit used by groff, is 1/72 of an inch.

‡Traditionally called "carding" or "feathering."

**I.e. the leading used in paragraphs of running text.

††Shimming and flex-spacing are mutually exclusive.

Should users not want *mom*'s default shimming of document element tags, it may be disabled and re-invoked at any time. Additionally, inserted images and preprocessor blocks have an option to disable it selectively.

FLEX-SPACING

When flex-spacing is enabled, *mom* divides any whitespace remaining before the bottom margin and distributes it equally at sensible flex-points on the page, e.g. after headings or graphics. The result, as with shimming, is equal bottom margins from page-to-page. In many instances, the visible results of shimming and flex-spacing are indistinguishable. The advantage to flex-spacing is that it compensates fully in documents where the paragraphs are separated by a small amount of whitespace; shimming cannot guarantee this unless the total amount of paragraph spacing on a page equals a multiple of the leading in effect.

Flex-points may be inserted by the user with the '**.FLEX**' macro, typically to improve page rhythm between selected, spaced paragraphs. Flex-spacing may also be disabled globally and re-invoked at any time, or disabled selectively for individual graphics or preprocessor blocks.

▪ Direct to PDF output

A number of *mom*'s features are only available when the source document is intended for PDF output: internal and external links, relocatable Table of Contents, flex-spacing, and others. For this reason, *mom* documents should be processed with the helper script, **pdfmom**. **Pdfmom** takes care of all processing required for direct to PDF output of *mom* documents, relieving users of the need to construct complex command-line invocations of groff.

Pdfmom accepts all the same command-line options as groff, including **-Tps** for documents where PostScript output is preferred, minus *mom*'s PDF features and flex-spacing.

---

‡‡Both forms move up on the page by one line.

Appendix

The stylesheet and source file for this document offer examples of typical and advanced groff and *mom* usage.

The stylesheet (groff-mom-style.mom)

The Plantin family used in the text was chosen not only for readability, but because the roman font exhibits two peculiarities: the keyboard apostrophe isn't mapped correctly to the close-quote character, and the font has no **fi** ligature. It is sometimes simpler to fix small font problems using groff requests rather than altering the font files themselves. Thus, at the top of the stylesheet file, the groff '`.char`' request is used to map the apostrophe correctly, and to create the **fi** ligature out of an **f** followed by a dotless-**i**.

The '`.char`' request is also used extensively to contain the formatting for emphasized words, such that when the words appear in the source file, they are unemcumbered by inline escapes and appear cleanly as `\[word]`.

The cover, docheader, epigraph, endnotes, and table of contents sections of the stylesheet demonstrate the use of control macros and style-groups to design various parts of a document.

The three levels of headings are given their own macros. Only the second level, SUBHEAD, requires special treatment because of the square bullet, but the other two wrap first and third level headings inside HEAD and SUBSUBHEAD so that consistency is achieved with respect to semantic tagging.

The source file (groff-mom.mom)

The use of a comprehensive stylesheet allows the text and document element tags to flow sensibly and readably without undue formatting interruptions. Presentational markup is restricted to track and character-pair kerning, which were added after previewing the completed document. Kernpairs can be adjusted in the font files themselves, but this is not always desirable. Track kerning is used to tighten or loosen paragraphs for optimal word breaks and to avoid widows and orphans.

It is worth noting that the relative absence of presentational markup makes the file parsable for semantic elements and thus, with little difficulty, convertible to other formats, e.g. xml or html.

Endnotes

[1] Macro examples are taken from the *mom* macro package.

[2] See '**info groff**' for documentation.

[3] Correctly, **\n[.l]u/2u**. Many groff registers, including **.l**, store their values in "machine units", typically 1/1000 of a PostScript point; this must be made explicit when using number registers as arguments to macros and requests by appending the scaling indicator '**u**'.

[4] Also *man*, which is used to format Unix manpages.

[5] **man *preprocessor*** provides complete documentation for any preprocessor.

[6] Limited support for html output is also supported.

[7] Use of the typesetting macros in conjunction with document processing is covered **here** in *mom*'s html documentation.

[8] When an absolute value is given, autoleading is equivalent to the hot-metal notion of expressing leading in terms of the number of strips of lead between lines of galley type. '**.AUTOLEAD 3**' means "three points of lead between every line, no matter what the point size."

[9] Autoleading in conjunction with the document processing macros is covered **here** in *mom*'s html documentation.