Using *groff* with the *ms* Macro Package

Larry Kollar

kollar@alltel.net

This document describes the GNU *roff* re-implementation of the popular *ms* macro package.

The *ms* macros are suitable for reports, letters, memoranda, books, user manuals, and other works. The package provides macros for cover page and table of contents generation, section headings, multiple paragraph styles, text styling (including font changes), lists, footnotes, pagination control, and indexing.

ms supports the *tbl*, *eqn*, *pic*, and *refer* preprocessors for inclusion of tables, mathematical equations, diagrams, and standardized bibliographic citations.

1. Introduction

The *ms* macros are the oldest surviving macro package for *roff* systems.[1] While the *man* package was designed for brief documents to be perused at a terminal, the *ms* macros are suitable for longer documents intended for printing and possible publication.

The *ms* macro package included as part of *groff* is a complete re-implementation. Some macros specific to AT&T or Berkeley are not included, while several new commands have been introduced. Experienced *ms* users may wish to skip ahead to section "Differences from AT&T *ms*" below.

If you're in a hurry to get started, you need only know that *ms* needs one of its macros called at the beginning of a document so that it can initialize. A paragraph macro like `.PP` (if you want your paragraph to have a first-line indent) or `.LP` (if you don't) suffices.

After that, start typing normally. You can separate paragraphs with further paragraph macros, or with blank lines, and you can indent with tabs. When you need one of the features mentioned in the abstract above, return to this document.

An example of a simple but complete *ms* document follows.

```
.LP
Radical novelties are so disturbing that they tend to be
suppressed or ignored, to the extent that even the
possibility of their existence in general is more often
denied than admitted.

→That's what Dijkstra said, anyway (in EWD 1036).
```

We have used an arrow (→) in the above to indicate a tab character.

[1] Although man *pages* are even older, the *man* macro language dates back only to Seventh Edition Unix (1979). *ms* was documented by Michael E. Lesk in an article for the journal *Communications of the ACM* in 1974.

1.1. Registers, strings, and units

ms exposes many aspects of document layout to user control via *groff* registers and strings. To use them, you must understand how to define them with *requests*, which look just like *ms* macro calls, but use lowercase letters. Requests and macro calls always appear on input lines by themselves starting with a dot.[2]

`.nr` *reg* [*value*]
Set register *reg* to *value*. If *value* is omitted, *groff* defines *reg* as 0.

`.ds` *name* [*contents*]
Set string *name* to *contents*. If *contents* is omitted, *groff* defines *name* as empty.

Most registers have a default unit of measurement. Specifying a unit explicitly does not cause any problems, and can avoid problems in complex situations. The following are typical units.

Unit	Description
`i`	inches
`c`	centimeters
`p`	points (about 1/72″)
`v`	"vees"; line height
`n`	"ens"; width of a letter "n" in the current font
`m`	"ems"; width of a letter "m" in the current font

The counterpart of storage is retrieval. Registers and strings are set with requests, but retrieved using escape sequences in a process called *interpolation*.[3] Interpolation replaces the escape sequence in the output.

`\n[`*reg*`]`
Interpolate the value of register *reg*.

`\*[`*name*`]`
Interpolate the contents of string *name*.

In this document, we use the interpolation syntax for registers and strings to refer to them except where they are otherwise made clear by context (such as a column heading in a table); similarly, we generally precede the names of macros and requests with a dot. These are frequent practices in *roff* documentation to prevent reader confusion. You would never write "`.nr \n[PS] 12`" or "`.ds \*[FAM] T`" unless you're trying to do something frightfully clever and beyond the scope of this discussion.

You can define registers and strings that *ms* knows nothing about, and use them for your own purposes. If you do, be mindful of the name spaces for registers and other names; see "Naming conventions" below. A good habit to cultivate with strings is to end each definition with a comment escape, `\"`, because strings are allowed to contain spaces and tabs, even at the end. If your text editor does not show you such trailing white space, it can cause much frustration.

`.ds dog Spot→` *(accidental trailing tab)*
`See \*[dog].`
`See \*[dog] run.`
	See Spot . See Spot run.

`.ds dog Spot\"→` *(another trailing tab)*
`Run, \*[dog], run!`
`Good boy.`
	Run, Spot, run! Good boy.

[2] Experienced *roff*ers can tell you of exceptions involving conditional requests and the configurability of the control characters—and the escape character, too, since we're about to get to that.

[3] "Expansion" is another commonly used term, but can be misleading; particularly in the case of registers, the replacement text can be shorter than the escape sequence that produced it.

2. Structure of an *ms* document

A simple document was seen in the introduction; longer ones have a structure as follows.

Document format

Historically, sites developed templates embedded in macros to be called at the beginning of a document; this would set up page margins, headers, footers, and other features that can differ from the *ms* defaults. The only such format originally fully documented was the *report*. If you invoke the `.RP` macro on the first line of the document, *ms* prints the document description information (see below) on its own page, the *cover page*; otherwise it prints the information (if any) on the first page with your document text immediately following. Some document types found in other *troff* implementations are specific to AT&T or Berkeley, and are not supported in *groff ms*.

Format and layout

By setting registers (and one string), you can change your document's typeface (font and point size), margins, spacing (leading), headers and footers, and footnote properties. See "Document control settings" below.

Document description

A document description consists of a title, the name and affiliated institution of one or more authors, an abstract, and a date.[4] See "Document description macros" below.

Body

Following the cover page is your document. *ms* supports highly structured documents, consisting of paragraphs interspersed with multi-level headings (chapters, sections, subsections, and so forth) and augmented by lists, footnotes, tables, diagrams, and other constructs. See "Text" below for more details.

Table of contents

Longer documents include a table of contents, which you can produce by placing the `.TC` macro at the end of your document. Emitting the table of contents at the end is necessary since GNU *troff*, like its AT&T ancestor, is a single-pass formatter; it thus cannot determine the page number of a section until that section has actually been set and output. Since *ms* output is designed for hard copy, you can manually relocate the pages containing the table of contents between the cover page and the body text after printing.[5]

[4] Actually, only the title is required.

[5] This limitation could also be overcome by using PostScript or PDF file manipulation utilities to resequence pages in the document, facilitated by specially-formatted comments ("device tags") placed in the output by by *ms*.

3. Document control settings

For consistency, set registers related to margins at the beginning of your document, or just after the `.RP` macro. You can set other registers later in your document, but you should keep them together at the beginning to make them easy to find and edit as necessary.

The following table summarizes the available settings for convenience; each is explained in greater detail in an appropriate section below.

Type	Register	Definition	Effective	Default
Margins	`PO`	Page offset (left margin)	next page	`1i`
	`LL`	Line length	next paragraph	`6i`
	`LT`	Header/footer length	next paragraph	`\n[LL]`
	`HM`	Top (header) margin	next page	`1i`
	`FM`	Bottom (footer) margin	next page	`1i`
Text	`PS`	Point size	next paragraph	`10p`
	`VS`	Vertical spacing	next paragraph	`\n[PS]+2`
	`PSINCR`	Heading size increment	next section	`1p`
	`GROWPS`	Heading level threshold	next section	`0`
	`HY`	Hyphenation mode	next paragraph	`6`
Paragraphs	`PI`	First-line indent	next paragraph	`5n`
	`PD`	Space between paragraphs	next paragraph	`0.3v`
	`QI`	Quoted paragraph indent	next paragraph	`5n`
	`PORPHANS`	lines kept in paragraph	next paragraph	`1`
	`HORPHANS`	lines kept with heading	next paragraph	`1`
Footnotes	`FL`	Length	next footnote	`\n[LL]*5/6`
	`FI`	Indent	next footnote	`2n`
	`FF`	Format	next footnote	`0`
	`FPS`	Point size	next footnote	`\n[PS]-2`
	`FVS`	Vertical spacing	next footnote	`\n[FPS]+2`
	`FPD`	Paragraph spacing	next footnote	`\n[PD]/2`
Other	`DD`	Display distance	next paragraph	`0.5v`
	`MINGW`	Minimum gutter width	next page	`2n`
Type	**String**	**Definition**	**Effective**	**Default**
Text	`FAM`	Font family	next paragraph	*varies*

Some settings' defaults are computed based on others; for instance, the expression for the default of `\n[FL]` means that the footnote length is five sixths of the line length. The default font family is dependent on the output driver; for typesetter devices, it is frequently Times. On terminals, *groff* has no control over the font family.

4. Document description macros

All but the simplest documents bear a title. As their level of sophistication (or complexity) increases, they tend to acquire dates of revision, explicitly identified authors, sponsoring institutions for authors, and, at the rarefied heights, an abstract of their content.

By default, *ms* places any such defined information on the first page of the document, followed by the first section heading or paragraph. Use the macros below in the order shown to define these data. Strictly, `.TL` is optional, in that a document need not have a title; but if you use any of the other macros shown here, it is mandatory.

Macro	Description
`.RP` [`no`]	(optional) Characterize the document as a "report". The report format creates a separate cover page. With the `no` argument, *ms* produces a cover page but does not repeat any of its information on the first page.
`.TL`	Specify the document title. *ms* collects text on input lines following this macro into the title until reaching an `.AU`, `.AB`, or sectioning or paragraph macro.
`.DA` [*xxx*]	(optional) Put the current date, or the arguments to the macro if specified, on the cover page (if present) *and* in the footers. This is the default for *nroff*.
`.ND` [*xxx*]	(optional) Put the current date, or the arguments to the macro if specified, on the cover page (if present) *but not* in the footers. This is the default for *troff*.
`.AU`	(optional) Specify an author's name. *ms* collects text on input lines following this macro into the author's name until reaching an `.AI`, `.AB`, or sectioning or paragraph macro.
`.AI`	(optional) Specify the preceding author's institution. An `.AU` macro is usefully followed by at most one `.AI` macro; if there are more, the last `.AI` controls. *ms* collects text on input lines following this macro into the author's institution until reaching an `.AU`, `.AB`, or sectioning or paragraph macro.
`.AB` [`no`]	(optional) Begin the abstract. *ms* collects text on input lines following this macro into the abstract until reaching an `.AE` macro. By default, *ms* places the word "ABSTRACT" centered and in italics above the text of the abstract. The argument `no` suppresses this heading.
`.AE`	(mandatory if `.AB` used; prohibited otherwise) End the abstract.

The following is example markup for a cover page.

```
.RP
.TL
The Inevitability of Code Bloat
in Commercial and Free Software
.AU
J. Random Luser
.AI
University of West Bumblefuzz
.AB
This report examines the long-term growth
of the code bases in two large, popular software
packages; the free Emacs and the commercial
Microsoft Word.
While differences appear in the type or order
of features added, due to the different
methodologies used, the results are the same
in the end.
.PP
The free software approach is shown to be
superior in that while free software can
become as bloated as commercial offerings,
free software tends to have fewer serious
bugs and the added features are in line with
user demand.
.AE
```

. . . the rest of the paper follows . . .

5. Text

We now turn to macros and registers used to structure and present the body of your document. Examples of such organization include paragraphs and sections. Let us begin, however, with more fundamental units of language and typography; characters, glyphs, and words. A *character* is an abstract symbol; we use the term to refer to the elements of input you give *ms*. A *glyph* is a rendered grapheme with properties like size, weight (stroke thickness), slant, serifs, and so on. Thus A, *A*, and **A** are all the same character, but distinguishable glyphs. Glyphs put ink on a page (real or virtual); spaces and tabs are characters in input, but merely gaps between glyphs in output.

To *groff*, a *word* is a sequence of characters that produce glyphs separated by sequences of characters that don't. The formatter does not comprehend natural languages, so to it, `mother-in-law`, `dog*star`, and ←`span`→ are each one word, and each includes all of the symbols shown. Understanding this will help you predict how *groff* performs breaking, hyphenation, and adjustment.

5.1. Size, vertical spacing, and hyphenation

Every glyph *ms* renders is typeset at a point size upon a lattice called the vertical spacing. The vertical spacing is also measured in points, and is conventionally chosen to fit the height of the glyphs in the font at its current point size, plus extra space termed *leading*. This word is pronounced to rhyme with "sledding", and refers to the use of lead metal (Latin: *plumbum*) in traditional typesetting.

In this example, the point size and vertical spacing are equal. Typographers would call this "11 on 11".
Jackdaws love my big sphinx of quartz. How vexingly quick daft zebras jump!

Meanwhile, in the horizontal dimension, we face the challenge of hyphenation. *ms* itself has no awareness of hyphenation; the formatter (*groff*) takes care of that.[6] *ms*'s role is to expose a register for setting the hyphenation mode, and clean up periodically by restoring that mode in what is called a "paragraph reset".

Register	Mnemonic	Effective	Default	Description
`PS`	Point size	next paragraph	`10p`	point size of body text
`VS`	Vertical spacing	next paragraph	`12p`	spacing between lines
`HY`	Hyphenation mode	next paragraph	`6`	hyphenation mode

As a *groff ms* extension, if `\n[PS]` or `\n[VS]` is set greater than or equal to 1000, *ms* divides the register by 1000 to get a fractional point size. For example, `.nr PS 1025` sets the point size to `10.25p`.

5.1.1. Superscripting and subscripting

Text enclosed with `\*{` and `\*}` is set as a superscript; thus `\*{238\*}U` produces "^{238}U". This example also illustrates a shorthand for string names of one character; you can write either `\*{` or `\*[{]`, for instance. Subscripts are similar; `C\*<8\*>H\*<18\*>` produces "C_8H_{18}".

String	Description
`\*[{]`	Begin superscripted text.
`\*[}]`	End superscripted text.
`\*[<]`	Begin subscripted text.
`\*[>]`	End subscripted text.

[6] See "Maniplating Hyphenation" in *groff: The GNU Implementation of Troff*, or section "Hyphenation" in *groff*(7).

5.1.2. Enlarging and reducing text

Three macros exist to alter the point size without changing the vertical spacing. Used sparingly, they can enable emphasis or understatement without disarranging the page in the vertical dimension.

Macro	Description
`.LG`	Set all subsequent text in larger type (2 points larger than the current point size) until the next point size, font style, paragraph, or heading macro. You can specify this macro multiple times to enlarge the point size as needed.
`.SM`	Set all subsequent text in smaller type (2 points smaller than the current point size) until the next point size, highlighting, paragraph, or heading macro. You can specify this macro multiple times to reduce the point size as needed.
`.NL`	Set all subsequent text at the normal point size (that is, the value of `\n[PS]`).

5.2. Changing font styles and families

The *ms* macros provide a variety of methods to decorate, highlight, or emphasize text. An important one is the selection of font. *Font* is a broad term that covers many parameters of a typeface, even apart from the point size. Fonts possess differing text *styles* and are often grouped into named *families*. Four styles are commonplace: roman, **bold**, *italics* (or oblique or slanted), and ***bold and italics combined***.

Macro	Description
`.B [`*`word`* `[`*`post`* `[`*`pre`*`]]]`	With no arguments, set text on subsequent input lines in **bold** until the next font, paragraph, or heading macro. Otherwise, set only *word* in **bold**. *post* is set after the bold text, in the previous typeface, with no intervening space; this is a Berkeley extension. Similarly, *pre* is set in the previous typeface *before* the first argument; this is a *groff* extension.
`.R [`*`word`* `[`*`post`* `[`*`pre`*`]]]`	As `.B`, but style text in roman instead of bold.
`.I [`*`word`* `[`*`post`* `[`*`pre`*`]]]`	As `.B`, but style text in *italics* instead of bold.
`.BI [`*`word`* `[`*`post`* `[`*`pre`*`]]]`	As `.B`, but style text in ***bold italics*** instead of bold.
`.CW [`*`word`* `[`*`post`* `[`*`pre`*`]]]`	As `.B`, but style text in a `constant-width typeface` instead of bold. This macro is a Berkeley extension.

GBR Discuss argument quotation. For example, "`.B foo ) (`" prints (**foo**).

GBR Talk about FAM and .fam here.

GBR Contrast decoration of text with font parameters.

Macro	Description
`.BX` *`word`*	Set *word* with a box around it. If you want to box a string that contains spaces, use a digit-width space escape, `\0` (backslash-zero).
`.UL` *`word`* `[`*`post`*`]`	Set *word* underlined. *post* is set without underlining immediately afterward, with no intervening space.

GBR `.B1 .B2`

GBR Forward reference to "Displays and keeps" for `PORPHANS`.

5.3. Paragraphs

Use the `.PP` macro to create a paragraph with a first-line indent (like the next paragraph), and the `.LP` macro to create a paragraph with no first-line indent (like this one).

The `.QP` macro indents its text at both left and right margins. The next paragraph or heading returns margins to normal.

The following example uses all three paragraph macros. GBR MISSING: `.XP`, `.QS`, `.QE`

```
.LP
The following software and versions were
considered for this report.
.PP
For commercial software, we chose
.B "Microsoft Word for Windows" ,
starting with version 1.0 through
Word 2000.
.PP
For free software, we chose
.B Emacs ,
from its first appearance as a standalone
editor through version 20.
.QP
Franklin's Law applied to software:
software expands to outgrow both
RAM and disk space over time.
```

5.4. Lists

The `.IP` macro handles duties for all lists. Its syntax is as follows:

`.IP` [*marker* [*width*]]

The *marker* is usually a bullet character (`\[bu]`) for unordered lists, a number (or auto-incrementing register) for numbered lists, or a word or phrase for indented (glossary-style) lists.

The *width* specifies the indent for the body of each list item. Once specified, the indent remains the same for all list items in the document until specified again.

The following are examples of each type of list.

Source	Result

`A bulleted list:`
`.IP \[bu] 2`
`lawyers`
`.IP \[bu]`
`guns`
`.IP \[bu]`
`money`
	A bulleted list:
	• lawyers
	• guns
	• money

`.nr step 1 1`
`A numbered list:`
`.IP \n[step] 3`
`lawyers`
`.IP \n+[step]`
`guns`
`.IP \n+[step]`
`money`
	A numbered list:
	1. lawyers
	2. guns
	3. money
	Note the use of the auto-incrementing number register in this example.

`A glossary-style list:`
`.IP lawyers 0.4i`
`Two or more attorneys.`
`.IP guns`
`Firearms, preferably`
`large-caliber.`
`.IP money`
`Gotta pay for those`
`lawyers and guns!`
	A glossary-style list:
	lawyers
	Two or more attorneys.
	guns Firearms, preferably large-caliber.
	money
	Gotta pay for those lawyers and guns!

GBR Introduce auto-incrementing register concept.

In the last example, note how the `.IP` macro places the definition on the same line as the term if it has enough space. This may or may not be the effect you want. The following example shows two possible workarounds.

Code	Result

`A glossary-style list:`
`.IP lawyers 0.4i`
`Two or more attorneys.`
`.IP guns`
`.br`
`Firearms, preferably`
`large-caliber.`
`.IP money`
`Gotta pay for those`
`lawyers and guns!`
	A glossary-style list:
	lawyers
	Two or more attorneys.
	guns
	Firearms, preferably large-caliber.
	money
	Gotta pay for those lawyers and guns!

`A glossary-style list:`
`.IP lawyers 0.4i`
`Two or more attorneys.`
`.IP guns\h'0.4i'`
`Firearms, preferably`
`large-caliber.`
`.IP money`
`Gotta pay for those`
`lawyers and guns!`
	A glossary-style list:
	lawyers
	Two or more attorneys.
	guns
	Firearms, preferably large-caliber.
	money
	Gotta pay for those lawyers and guns!

The first example uses the `.br` request to force a break after printing the term or label. The second example uses the `\h` escape to do the same thing, inserting a horizontal motion (which is not breakable space). We used the a motion of the same size as the indentation to guarantee that "guns" would be too wide to fit, without having to do any tedious measuring.

To set nested lists, use the relative inset macros, `.RS` and `.RE`. These macros begin and end a region indented to line up with the body of an `.IP` macro. For example:

`.IP \[bu] 2`
`Lawyers:`
`.RS`
`.IP \[bu]`
`Dewey,`
`.IP \[bu]`
`Cheatham,`
`.IP \[bu]`
`and Howe.`
`.RE`
`.IP \[bu]`
`Guns`
`...`
	• Lawyers:
	  • Dewey,
	  • Cheatham,
	  • and Howe.
	• Guns . . .

GBR Table with `.IP .RS .RE`

5.5. Footnotes

The *ms* macro package has a flexible footnote system. You can specify a numbered footnote[7] by using the `\**` escape, followed by the text of the footnote enclosed by `.FS` and `.FE` macros.

[7] This is a numbered footnote.

You can specify symbolic footnotes† by placing the character (such as `\[dg]` for the dagger character used here), followed by a pair of `.FS` and `.FE` macros enclosing the same character again and the footnote text. (In other words, you have to match the symbols yourself, whereas numbered footnotes are synchronized for you.[8])

You can control how *groff* prints footnote numbers at the page bottom by changing the value of the `FF` register as follows.

Value	Description
0	Prints the footnote number as a superscript; indents the footnote (default).
1	Prints the number followed by a period (e.g., "1.") and indents the footnote.
2	Like 1, without an indent.
3	Like 1, but prints the footnote number as a hanging paragraph.

5.6. Headings

Use headings to create a hierarchical structure for your document. The *ms* macros print headings in **bold**, using the same font family and point size as the body text.

The following table describes the heading macros. Text on input lines after these macros gives the section a title; any other macro ends the title. GBR fact-check this

Macro	Description
`.NH [`*`level`*`]`	Set a numbered heading. The argument *level* is number that indicates the level of the heading, The section headings in this document use the `.NH` macro to show the level of each section. If you skip a heading level, with `.NH 3` immediately after `.NH 1` for instance, *groff* writes a warning to the standard error output.
`.NH S` *n* `[...]`	Set an explicitly-numbered heading. Each argument *n* is a component of a multi-part section (subsection, subsubsection, etc.) number. Thus `.NH S 5 4 2` numbers the following section "5.4.2".
`.SH [`*`match-level`*`]`	Set an unnumbered heading. The optional *match-level* argument is a *groff ms* extension. Its purpose is to match the point size at which the heading is printed to the size of a numbered heading at the same level when the `\n[GROWPS]` and `\n[PSINCR]` heading size adjustment mechanism is in effect; see "Document control settings" above.

GBR Import `SN-DOT`, `SN-NO-DOT`, `SN-STYLE` from Texinfo.

GBR Forward reference to "Displays and keeps" for `HORPHANS`.

5.7. Displays and keeps

Use displays to show text-based examples or figures (such as code listings). This document shows *groff* code examples inside displays, for example.

Displays turn off filling, so lines of code can be displayed as-is without inserting `.br` requests in between each line. Displays can be *kept* on a single page, or allowed to break across pages. The following table shows the display types available.

†This is a symbolic footnote.

[8] If you guessed that an auto-incrementing register would be a good way to implement footnote numbers, macro programming in *groff* may lie in your future.

Display macro		Description
With keep	**No keep**	
`.DS L`	`.LD`	Left-justify display.
`.DS I [`*indent*`]`	`.ID [`*indent*`]`	Indent display by *indent* (default: `\n[DI]`).
`.DS B`	`.BD`	Block-center display (left-justify, longest line centered).
`.DS C`	`.CD`	Center all lines in display.
`.DS R`	`.RD`	Right-justify all lines in display.

Use the `.DE` macro to end any display type.

On occasion, you may want to *keep* other text together on a page. For example, you may want to keep two paragraphs together, or a paragraph that refers to a table (or list, or other item) immediately following. The *ms* macros provide the `.KS` and `.KE` macros for this purpose. The `.KS` macro begins a block of text to be kept on a single page, and the `.KE` macro ends the block.

You can specify a *floating keep*; if the keep cannot fit on the current page, *groff* holds the contents of the keep and allows text following the keep (in the source file) to fill in the remainder of the current page. When the page breaks, whether by an explicit `.bp` request or by reaching the end of the page, *groff* prints the floating keep at the top of the new page. This is useful for printing large graphics or tables that do not need to appear exactly where specified. Use the `.KF` and `.KE` macros to specify a floating keep.

You can also use the "need" request, `.ne` *n*, to force a page break if there is not enough vertical space remaining on the page to set *n* lines of text together at the current point size and vertical spacing.

5.8. Tables, figures, equations, and references

Mark text meant for a preprocessor by enclosing it in a pair of tags as follows.

Tag Pair	Description
`.TS [H]` `.TE`	Denotes a table to be processed by the *tbl* preprocessor. The optional **H** argument to `.TS` instructs *groff* to create a running header with the information up to the `.TH` macro. *groff* prints the header at the beginning of the table; if the table runs onto another page, *groff* prints the header on the next page as well.
`.PS` `.PE`	Denotes a graphic to be processed by the *pic* preprocessor. You can create a *pic* file by hand, using the AT&T *pic* manual available on the Web as a reference, or by using a graphics program such as *xfig*.
`.EQ [`*align*`]` `.EN`	Denotes an equation to be processed by the *eqn* preprocessor. The optional *align* argument can be **C**, **L**, or **I** to center (the default), left-justify, or indent the equation.
`.[` `.]`	Denotes a reference to be processed by the *refer* preprocessor. The *refer*(1) man page provides a comprehensive reference to the preprocessor and the format of the bibliographic database.

5.8.1. An example multi-page table

The following is an example of how to set up a table that may print across two or more pages.

```
.TS H
allbox expand;
cb | cb .
Text...of heading...
_
.TH
.T&
l | l .
...the rest of the table follows...
.TE
```

6. Page layout

The default output from the *ms* macros provides a minimalist page layout: it prints a single column, with the page number centered at the top of each page. It prints no footers.

You can change the layout by setting the proper registers and strings.

6.1. Headers and footers

There are two ways to define headers and footers:

- Set the strings `LH`, `CH`, and `RH`, to set the left, center, and right headers; and `LF`, `CF`, and `RF` to set the left, center, and right footers. This works best for documents that do not distinguish between odd and even pages.
- Use the `.OH` and `.EH` macros to define headers for the odd and even pages; and `.OF` and `.EF` macros to define footers for the odd and even pages. This is more flexible than defining the individual strings. The syntax for these macros is as follows:

  ```
  .xH 'left'center'right'
  ```

  where *x* is `E` or `O` for even- or odd-numbered pages, respectively. You can replace the quote (') marks with any character not appearing in the header or footer text.

  GBR Do the above better with tables.

6.2. Margins

Control margins with registers. The following table lists the register names and defaults.

Register	Definition	Effective	Default
`PO`	Page offset (left margin)	next page	`1i`
`LL`	Line length	next paragraph	`6i`
`LT`	Length of titles (headers, footers)	next paragraph	`\n[LL]`
`HM`	Header margin (top margin)	next page	`1i`
`FM`	Footer margin (bottom margin)	next page	`1i`

`PO` defines the page offset, also known as the left margin, and `LL` the line length, or width of the body text. There is no right margin setting; the combination of page offset and line length provides the information necessary to derive the right margin.

`LT` sets the length of "titles", a *roff* term for headers and footers regarded together; it has nothing to do with a document title.[9] `HM` and `FM` define the header margin (at the page top) and the footer margin (at the page bottom), respectively. They apply even if a page's corresponding header or footer is empty.

[9] Except insofar as a document title is a reasonable choice for header or footer content.

6.3. Multiple columns

The *ms* macros can set text in as many columns as will reasonably fit on the page. The following macros are available. All of them force a page break if a multi-column layout is already active. However, if the current mode is single-column, starting a multi-column layout does *not* force a page break.

Macro	Description
`.1C`	Single-column mode.
`.2C`	Two-column mode.
`.MC` [*width* [*gutter*]]	Multi-column mode. If you specify no arguments, it is equivalent to the `.2C` macro. Otherwise, *width* is the width of each column and *gutter* is the space between columns. The `MINGW` register is the default gutter width.

7. Safe *roff* requests

GBR Write me.

8. Creating a table of contents

The facilities in the *ms* macro package for creating a table of contents are semi-automated at best. Assuming that you want the table of contents to consist of the document's headings, you need to repeat those headings wrapped in `.XS` and `.XE` macros.

In addition, the `.XS` macro does not know to indent a heading based on its level. The easiest way to work around this is to add tabs to the table of contents string. The following is an example:

```
.NH 1
Introduction
.XS
Introduction
.XE
...
.NH 2
Methodology
.XS
    Methodology
.XE
...
```

GBR Add table, document `.XA`.

The *Groff and Friends HOWTO* includes a *sed* script that automatically inserts `.XS` and `.XE` entries after each heading in a document.

Altering the `.NH` macro to automatically build the table of contents is perhaps initially more difficult, but would save a great deal of time in the long run if you use *ms* regularly.

9. Creating an index

GBR Document `.IX` here.

Macro	Description
`.IX`	Indexing term (written to the standard error stream).

10. Naming conventions

groff has one name space for registers, and another for macros, strings, and other objects an *ms* user normally need not worry about;[10] the latter category is referred to simply as a "name" for lack of a better categorical term. So the `\n[CW]` register is distinct from the `\*[CW]` string, but the `\*[CW]` string uses the same storage as the `.CW` macro. Therefore, do not define a string `PP` to store the name of, say, a political party unless you're comfortable with it clobbering the `.PP` macro and rendering it useless. Even if you don't use a certain *ms* macro yourself in a given document, you cannot assume its name is free for the taking; *ms* macros call each other behind the scenes.

The *groff ms* macro package reserves the following identifiers for both registers and names.

- sequences containing the characters `*`, `@`, `:`, and `!`
- sequences containing only uppercase letters and digits

In other words, mixed-case registers and names lacking the characters `*@:!` are safe for your own use; *groff ms* will not use them.

11. *groff ms* internals

Those needing to troubleshoot the *groff ms* macro package may wish to be aware of some aspects of its design. In the ensuing discussion, and *identifier* is either a *reg* or a *name*; see "Registers, strings, and units" above.

- Identifiers intended for use by documents contain only uppercase letters and digits.
- Internally, the macros are divided into modules.
- Identifiers used only within a module are of the form *`module*id`*.
- Identifiers used outside the module in which they are defined are of the form *`module@id`*.
- Identifiers associated with a *groff* environment are of the form *`module:id`*. These are used only within the `par` module at present.
- Identifiers constructed to implement arrays are of the form *`array!index`*.

[10] Such as diversions, about which the unlucky user with an ill-structured document may get diagnostic messages from *ms*.

12. Differences from AT&T *ms*

This section lists the (minor) differences between the *groff ms* macros and AT&T *troff ms* macros.

12.1. *troff* macros not appearing in *groff*

Most macros missing from *groff ms* are cover page macros specific to Bell Labs. The macros known to be missing are:

.TM Technical memorandum; a cover sheet style

.IM Internal memorandum; a cover sheet style

.MR Memorandum for record; a cover sheet style

.MF Memorandum for file; a cover sheet style

.EG Engineer's notes; a cover sheet style

.TR Computing Science Technical Report; a cover sheet style

.OK Other keywords

.CS Cover sheet information

.HO Bell Labs Holmdel postal address

.IH Bell Labs Naperville postal address

.MH Bell Labs Murray Hill postal address

.PY Bell Labs Piscataway postal address

.WH Bell Labs Whippany postal address

GBR Missing Berkeley macros: `.TM` thesis mode, `.CT` thesis chapter title

12.2. *groff* macros not appearing in AT&T *troff*

The *groff ms* macros have a few minor extensions compared to the AT&T *troff ms* macros.

.AM Improved accent marks (Berkeley).

.DS I Indented display. The default behavior of AT&T *troff ms* was to indent; the *groff* default prints displays flush left with the body text. GBR This is not missing; remove or relocate.

.CW Print text in `constant width` (Courier) font (Berkeley).

.IX Indexing term (written to the standard error stream) (Berkeley).

The `MINGW` register specifies a minimum space between columns (for multi-column output); this takes the place of the `GW` register that was documented but apparently not implemented in AT&T *troff*. GBR Wrong; get footnote from Texinfo. Several new string registers are available as well. You can change these to handle (for example) the local language.

REFERENCES
: Contains the string printed at the beginning of the references (bibliography) page. GBR But nothing ever creates a bibiliography page to interpolate this string?

ABSTRACT
: Contains the string printed at the beginning of the abstract.

TOC
: Contains the string printed at the beginning of the table of contents.

GBR `MONTH1-MONTH12 MO DY`

13. Acknowledgements

Two documents provided essential reference material:

- The “Groff and Friends HOWTO” by Dean Allen Provins.
- “Typing Documents on the Unix System: Using the -ms Macros with Troff and Nroff”, the original AT&T *ms* documentation by M. E. Lesk.

Without these documents close at hand, writing this document would have been a much more difficult task.

GBR Berkeley *ms* paper by Kies, Tuthill(?), Borst-Rothe, Heydt

14. Undocumented miscellaneous temporary section

GBR UNDOCUMENTED: `.FP .UX \n[GS] .RT .PT .BT .HD .TA .SH-NO-TAG .PX \n[H1]` ... and devtags support. The above should either be documented or moved into a module namespace. If we need them for compatibility, document them.

GBR The "Bell localisms" stuff seems bogus. The 1976 edition of Kernighan & Cherry either used undocumented macros or a post-Version 7 Unix version of the paper was used for testing. No `.P1` or `.UC` macros appear in either the V6 or V7 versions of *tmac.s*.

Table of Contents