Using *groff* with the *ms* Macro Package

Larry Kollar

kollar@alltel.net

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

The *ms* macros are suitable for reports, letters, books, user manuals, and so forth. The package provides macros for cover pages, section headings, paragraphs, lists, footnotes, pagination, and a table of contents.

This document provides both tutorial and reference material. It also discusses differences from the AT&T *ms* macros, both omissions and additions.

1. Introduction

The original *ms* macros were included with AT&T *troff* alongside the *man* macros. While the *man* package is intended for brief documents that can be read on-line as well as printed, the *ms* macros are suitable for longer documents that are meant to be printed rather than read on-line.

The *ms* macro package included with *groff* is a complete, bottom-up re-implementation. Several macros (specific to AT&T or Berkeley) are not included, while several new commands are. See *Differences from AT&T ms*, later in this document, for more information.

1.1. Basic information

ms exposes many aspects of document layout to user control via *groff* registers and strings. 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 table summarizes typical units.

Unit	Description
`i`	inches
`c`	centimeters
`p`	points (1/72")
`v`	"vees"; height of a line using the current font
`n`	"ens"; width of an "n" using the current font
`m`	"ems"; width of an "M" using the current font
`P`	picas (1/6")

Set registers with using the `.nr` request and strings with the `.ds` request.

```
.nr PS 12 \" Use 12-point type.
.ds FAM P \" Use Palatino font family.
```

2. General structure of an *-ms* document

The *-ms* macro package expects a certain amount of structure, but not as much as packages such as *-man* or *-mdoc*.

The simplest documents can begin with a paragraph macro (such as `.LP` or `.PP`), and consist of text separated by paragraph macros or even blank lines. Longer documents have a structure as follows:

Document type
: If you invoke the `.RP` (report) macro on the first line of the document, *groff* prints the cover page information on its own page; otherwise it prints the information on the first page with your document text immediately following. Other document formats found in AT&T *troff* are specific to AT&T or Berkeley, and are not supported in *groff*.

Format and layout
: By setting number registers, you can change your document's type (font and size), margins, spacing, headers and footers, and footnotes. See *Document control settings* below for more details.

Cover page
: A cover page consists of a title, the author's name and institution, an abstract, and the date.[1] See *Cover page macros* below for more details.

Body
: Following the cover page is your document. You can use the *-ms* macros to write reports, letters, books, and so forth. The package is designed for structured documents, consisting of paragraphs interspersed with headings and augmented by lists, footnotes, tables, and other common constructs. See *Body text* below for more details.

Table of contents
: Longer documents include a table of contents, which you can invoke by placing the `.TC` macro at the end of your document.[2] Printing the table of contents at the end is necessary since *groff* is a single-pass text formatter, thus it cannot determine the page number of each section until that section has actually been set and printed. Since *-ms* output was intended for hardcopy, the standard procedure was to manually relocate the pages containing the table of contents between the cover page and the body text.

---

[1] Actually, only the title is required.

[2] The *-ms* macros have minimal indexing facilities, consisting of the `.IX` macro, which prints an entry on standard error.

3. Document control settings

For consistent document rendering, set registers related to margins at the beginning of your document, or just after calling `.RP`. You can set other registers later in your document, but keeping them together at the beginning makes them easy to find and edit as necessary.

Type	Parameter	Definition	Effective	Default
Margins	`\n[PO]`	Page offset (left margin)	next page	`1i`
	`\n[LL]`	Line length	next ¶	`6i`
	`\n[LT]`	Header/footer (title) length	next ¶	`6i`
	`\n[HM]`	Top (header) margin	next page	`1i`
	`\n[FM]`	Bottom (footer) margin	next page	`1i`
Text	`\n[PS]`	Point size	next ¶	`10p`
	`\n[VS]`	Line spacing (leading)	next ¶	`12p`
	`\n[HY]`	Hyphenation mode	next ¶	`6`
	`\*[FAM]`	Font family	next ¶	`T`
Paragraphs	`\n[PI]`	Initial indent	next ¶	`5n`
	`\n[PD]`	Space between paragraphs	next ¶	`0.3v`
	`\n[QI]`	Quoted paragraph indent	next ¶	`5n`
	`\n[PORPHANS]`	# of initial lines kept	next ¶	`1`
Section headings	`\n[PSINCR]`	Point size increment	next heading	`1p`
	`\n[GROWPS]`	Size increase level limit	next heading	`0`
	`\n[HORPHANS]`	# of following lines kept	next heading	`1`
	`\*[SN-STYLE]`	Numbering style (alias)	next heading	`\*[SN-DOT]`
Footnotes	`\n[FI]`	Indentation	next footnote	`2n`
	`\n[FF]`	Format	next footnote	`0`
	`\n[FPS]`	Point size	next footnote	`\n[PS]-2`
	`\n[FVS]`	Vertical spacing	next footnote	`\n[FPS]+2`
	`\n[FPD]`	Paragraph spacing	next footnote	`\n[PD]/2`
	`\*[FR]`	Line length ratio	special	`11/12`
Other	`\n[DD]`	Display distance (spacing)	next ¶	`0.5v`
	`\n[MINGW]`	Minimum gutter width	next page	`2n`

The default footnote line length is 11/12ths of the normal line length for compatibility with the expectations of historical *ms* documents; you may wish to set `\*[FR]` to `1` to align with contemporary typesetting practices. In the past,[3] an `FL` register was used for the line length in footnotes; however, setting this register at document initialization time had no effect on the footnote line length in multi-column arrangements.

`\*[FR]` should be used in preference to `\n[FL]` in contemporary documents. The footnote line length is effectively computed as "`\n[`*column-width*`] * \*[FR]`". If an absolute footnote line length is required, recall that arithmetic expressions in *roff* languages are evaluated from left to right.

```
.ds FR 0+3i \" Set footnote line length to 3 inches.
```

Changes to the footnote length ratio `\*[FR]` take effect with the next footnote written in single-column arrangements, but on the next page in multiple-column contexts.

The `\n[DD]` distance used before and after each pair of display macros is also applied to regions of the document preprocessed with *eqn*(1), *pic*(1), and *tbl*(1). The "gutter" affected by `\n[MINGW]` is the gap between columns in multiple-column page arrangements.

---

[3] in Version 7 Unix *ms*, its descendants, and GNU *ms* prior to *groff* version 1.23.0

4. Cover page macros

Use the following macros to create a cover page for your document in the order shown.

<table>
<tr><th>Macro</th><th>Description</th></tr>
<tr><td><code>.RP [</code>no<code>]</code></td><td>Specifies the report format for your document. The report format creates a separate cover page. The default action (no <code>.RP</code> macro) is to print a subset of the cover page on page 1 of your document.

If you use the optional no argument, groff prints a title page but does not repeat any of the title page information (title, author, abstract, etc.) on page 1 of the document.</td></tr>
<tr><td><code>.DA [</code>xxx<code>]</code></td><td>(optional) Print the current date, or the arguments to the macro if any, on the title page (if specified) and in the footers. This is the default for nroff.</td></tr>
<tr><td><code>.ND xxx [</code>xxx<code>]</code></td><td>(optional) Print the current date, or the arguments to the macro if any, on the title page (if specified) but not in the footers. This is the default for troff.</td></tr>
<tr><td><code>.TL</code></td><td>Specifies the document title. groff collects text following the <code>.TL</code> macro into the title, until reaching the author name or abstract.</td></tr>
<tr><td><code>.AU</code></td><td>Specifies the author's name, which appears on the line (or lines) immediately following. You can specify multiple authors as follows.
<pre>.AU
John Doe
.AI
University of West Bumblefuzz
.AU
Martha Buck
.AI
Monolithic Corporation
...</pre></td></tr>
<tr><td><code>.AI</code></td><td>Specifies the author's institution. You can specify multiple institutions in the same way that you specify multiple authors.</td></tr>
<tr><td><code>.AB [no]</code></td><td>Begins the abstract. The default is to print the word ABSTRACT, centered and in italics, above the text of the abstract. The option <code>no</code> suppresses this heading.</td></tr>
<tr><td><code>.AE</code></td><td>End the abstract.</td></tr>
</table>

The following is example markup for a title 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. Body text

This section describes macros used to mark up the body of your document. Examples include paragraphs, sections, and other groups.

5.1. Paragraphs

Use the `.PP` macro to create indented paragraphs (like the next paragraph), and the `.LP` macro to create paragraphs with no initial indent (like this one).

The `.QP` macro indents its text at both left and right margins. The effect is identical to the HTML `<BLOCKQUOTE>` element. The next paragraph or heading returns margins to normal.

The following markup uses all three paragraph macros.

```
.NH 2
Cases used in the study
.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 the current version
(Word 2000).
.PP
For free software,
we chose
.B Emacs ,
from its first appearance as a standalone editor through the
current version (v20).
.QP
Franklin's Law applied to software:
software expands to outgrow both RAM and disk space over time.
```

5.2. Headings

Use headings to create a hierarchical structure for your document. The *ms* macros print headings in **bold** using the same font family and, by default, point size as the body text. Numbered and unnumbered section headings are available. Text lines after heading macros are treated as part of the heading, rendered on the same output line in the same style.

Macro	Description
`.NH` *level*	Numbered heading. The *level* argument instructs *ms* to number sections in the form *x.y.z*, to any depth desired, with the numbering of each level increasing automatically and being reset when a more significant level is increased. `1` is the most significant or coarsest division of the document. Only nonzero values are output. If you specify heading levels with a gap in an ascending sequence, such as by invoking "`.NH 3`" after "`.NH 1`", *groff ms* emits a warning on the standard error stream. Alternatively, a first argument of `S` can be given, followed by integral arguments to number the levels of the heading explicitly. Further automatic section numbering, if used, resumes using the specified section numbers as their predecessors.

An example may be illustrative.

Code	Output
`.NH 1` `Animalia`	**1. Animalia**
`.NH 2` `Arthropoda`	**1.1. Arthropoda**
`.NH 3` `Crustacea`	**1.1.1. Crustacea**
`.NH 2` `Chordata`	**1.2. Chordata**
`.NH S 6 6 6` `Daimonia`	**6.6.6. Daimonia**
`.NH 1` `Plantae`	**7. Plantae**

After invocation of `.NH`, the assigned section number is made available in the strings `SN-DOT` (as it appears in a printed section heading with default formatting, followed by a terminating period) and `SN-NO-DOT` (with the terminating period omitted).

You can control the style used to print section numbers within numbered section headings by defining an appropriate alias for the string `SN-STYLE`. By default, `\*[SN-STYLE]` is aliased to `\*[SN-DOT]`. If you prefer to omit the terminating period from section numbers appearing in numbered section headings, you may define the alias as follows.

```
.als SN-STYLE SN-NO-DOT
```

Any such change in section numbering style becomes effective from the next use of `.NH`, following redefinition of the alias for `\*[SN-STYLE]`.

Macro	Description
`.SH [`*level*`]`	Unnumbered heading. The optional *level* argument is a GNU extension indicating the heading level corresponding to the *level* argument of `.NH`. It matches the point size at which the heading is printed to that of a numbered heading at the same level when the `\n[GROWPS]` and `\n[PSINCR]` heading size adjustment mechanism is in effect.

If the `GROWPS` register is set to a value greater than the *level* argument to `.NH` or `.SH`, the point size of a heading produced by these macros increases by `\n[PSINCR]` units over the size specified by `\n[PS]`, multiplied by the difference between *level* and `\n[GROWPS]`. The value stored in `\n[PSINCR]` is interpreted in *groff* basic units; the "`p`" scaling factor should be employed when assigning a value specified in points. For example, the sequence

```
.nr PS 10
.nr GROWPS 3
.nr PSINCR 1.5p
.NH 1
Carnivora
.NH 2
Felinae
.NH 3
Felis catus
.SH 2
Machairodontinae
```

will cause "1. Carnivora" to be printed in 13-point text, followed by "1.1. Felinae" in 11.5-point text, while "1.1.1. Felis catus" and all more deeply nested heading levels will remain in the 10-point text specified by the `PS` register. "Machairodontinae" is printed at 11.5 points, since it corresponds to heading level 2.

The `\n[HORPHANS]` register operates in conjunction with the `NH` and `SH` macros to inhibit the printing of orphaned section headings at the bottom of a page; it specifies the minimum number of lines of the subsequent paragraph that must be kept on the same page as the heading. If insufficient space remains on the current page to accommodate the heading and this number of lines of paragraph text, a page break is forced before the heading is printed. Any display macro or *tbl*, *pic*, or *eqn* region between the heading and the subsequent paragraph suppresses this grouping.

5.3. Highlighting

The *-ms* macros provide a variety of methods to highlight or emphasize text:

Macro	Description
`.B [txt [post [pre]]]`	Sets its first argument in **bold type**. If you specify a second argument, *groff* prints it in the previous font after the bold text, with no intervening space (this allows you to set punctuation after the highlighted text without highlighting the punctuation). Similarly, it prints the third argument (if any) in the previous font **before** the first argument. For example, .B foo) (prints (**foo**). If you give this macro no arguments, *groff* prints all text following in bold until the next highlighting, paragraph, or heading macro.
`.R [txt [post [pre]]]`	Sets its first argument in roman (or regular) type. It operates similarly to the `.B` macro otherwise.
`.I [txt [post [pre]]]`	Sets its first argument in *italic type*. It operates similarly to the `.B` macro otherwise.
`.CW [txt [post [pre]]]`	Sets its first argument in a `constant width face`. It operates similarly to the `.B` macro otherwise.
`.BI [txt [post [pre]]]`	Sets its first argument in ***bold italic type***. It operates similarly to the `.B` macro otherwise.
`.BX [txt]`	Prints its argument and draws a box around it. If you want to box a string that contains spaces, use a digit-width space (\0).
`.UL [txt [post]]`	Prints its first argument with an underline. If you specify a second argument, *groff* prints it in the previous font after the underlined text, with no intervening space.
`.LG`	Prints all text following in larger type (2 points larger than the current point size) until the next font size, highlighting, paragraph, or heading macro. You can specify this macro multiple times to enlarge the point size as needed.
`.SM`	Prints all text following in smaller type (2 points smaller than the current point size) until the next type size, highlighting, paragraph, or heading macro. You can specify this macro multiple times to reduce the point size as needed.
`.NL`	Prints all text following in the normal point size (that is, the value of the `PS` register).

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 number 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

`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!

The second example sets up an auto-incrementing register, `step`. In the last example, observe how the `.IP` macro places the definition on the same line as the term if it has enough space. If this is not what you want, there are a couple of 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 `\p` escape to do the same thing. Note the space following the escape; this is important. If you omit the space, *groff* prints the first word on the same line as the term or label (if it fits) **then** breaks the line.

5.5. Indented regions

You may need to indent a section of text while still wrapping and filling.

Macro	Description
`.RS`	Begin a region indented by `\n[PI]`, affecting the placement of headings, paragraphs, and displays.
`.RE`	End the most recent indented region.

You can use `.RS`/ `.RE` regions to line up text under hanging and indented paragraphs. For example, you may wish to nest lists.

<pre>.IP \[bu] 2
Lawyers:
.RS
.IP \[bu]
Dewey,
.IP \[bu]
Cheatham,
and
.IP \[bu]
Howe.
.RE
.IP \[bu]
Guns
...</pre>
	• Lawyers:
	    • Dewey,
	    • Cheatham, and
	    • Howe.
	• Guns . . .

5.6. 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.

Display macro With keep	 No keep	Description
`.DS L`	`.LD`	Left-aligns display.
`.DS I` [*indent*]	`.ID`	Indents display (by *indent* if given, otherwise the value of `\n[DI]`).
`.DS B`	`.BD`	Block-centers display (left-aligns with the longest line centered).
`.DS C`	`.CD`	Centers all lines in the display.
`.DS R`	`.RD`	Right-aligns all lines in the 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 `.ne` request to force a page break if there is not enough vertical space remaining on the page.

5.7. Tables, figures, equations, and references

The *-ms* macros support the standard *groff* preprocessors: *tbl*, *pic*, *eqn*, and *refer*. You mark text meant for preprocessors by enclosing it in pairs 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 GNU *refer*(1) man page provides a comprehensive reference to the preprocessor and the format of the bibliographic database.

5.7.1. 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
```

The result?

Part	**Description**
1	The Inevitability of Code Bloat in Commercial and Free Software
2	The Inevitability of Code Bloat in Commercial and Free Software
3	The Inevitability of Code Bloat in Commercial and Free Software
4	The Inevitability of Code Bloat in Commercial and Free Software
5	The Inevitability of Code Bloat in Commercial and Free Software
6	The Inevitability of Code Bloat in Commercial and Free Software
7	The Inevitability of Code Bloat in Commercial and Free Software
8	The Inevitability of Code Bloat in Commercial and Free Software
9	The Inevitability of Code Bloat in Commercial and Free Software
10	The Inevitability of Code Bloat in Commercial and Free Software
11	The Inevitability of Code Bloat in Commercial and Free Software
12	The Inevitability of Code Bloat in Commercial and Free Software
13	The Inevitability of Code Bloat in Commercial and Free Software
14	The Inevitability of Code Bloat in Commercial and Free Software
15	The Inevitability of Code Bloat in Commercial and Free Software
16	The Inevitability of Code Bloat in Commercial and Free Software
17	The Inevitability of Code Bloat in Commercial and Free Software
18	The Inevitability of Code Bloat in Commercial and Free Software
19	The Inevitability of Code Bloat in Commercial and Free Software
20	The Inevitability of Code Bloat in Commercial and Free Software
21	The Inevitability of Code Bloat in Commercial and Free Software
22	The Inevitability of Code Bloat in Commercial and Free Software
23	The Inevitability of Code Bloat in Commercial and Free Software
24	The Inevitability of Code Bloat in Commercial and Free Software
25	The Inevitability of Code Bloat in Commercial and Free Software
26	The Inevitability of Code Bloat in Commercial and Free Software
27	The Inevitability of Code Bloat in Commercial and Free Software
28	The Inevitability of Code Bloat in Commercial and Free Software
29	The Inevitability of Code Bloat in Commercial and Free Software

30	The Inevitability of Code Bloat in Commercial and Free Software

5.8. Footnotes

The *ms* macros provide a flexible footnote system. You can specify a numbered footnote[4] by using the `\**` escape, followed by the text of the footnote enclosed by `.FS` and `.FE` macros.

You can specify symbolic footnotes† by placing the mark character (such as `\[dg]` for the dagger character used here), followed by the symbol and the text of the footnote enclosed by `.FS` and `.FE` macros.

You can control how *ms* prints footnote numbers by changing the value of the `FF` register as follows.

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

You can use footnotes safely within keeps and displays, but avoid using numbered footnotes within floating keeps. You can set a second `\**` between a `\**` and its corresponding `.FS`, as long as each `.FS` occurs *after* the corresponding `\**` and the occurrences of `.FS` are in the same order as the corresponding occurrences of `\**`.

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 number 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:

  `.OH '`*left*`'`*center*`'`*right*`'`

You can replace the neutral apostrophes (') with any character not appearing in the header or footer text.

6.2. Margins

Control margins using registers. These are summarized in the "Margins" section of the table in section "Document control settings" above. There is no right margin setting; the combination of page offset and line length provide the information necessary to derive the right margin.

---

[4] This is a numbered footnote.

†This is a symbolic footnote.

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 mode is already set. However, if the current mode is single-column, starting a multi-column mode 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` number register is the default gutter width.

6.4. 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
...
```

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.

7. Differences from AT&T *-ms*

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

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

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 Memo for record; a cover sheet style

.MF Memo for file; a cover sheet style

.EG Engineer's notes; a cover sheet style

.TR Computing Science Tech Report; a cover sheet style

.OK Other keywords

.CS Cover sheet information

.MH A cover sheet macro

7.2. *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.

.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.

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

.IX Indexing term (printed on standard error).

The `MINGW` number 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*. 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.

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

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

8. Acknowledgements

Two documents provided essential reference material:

- The *Groff and Friends HOWTO*, by Dean Allen Provins.
- *Using the -ms Macros with Troff and Nroff*, the original AT&T document by M. E. Lesk.

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

Table of Contents