Using *groff* with the *ms* Macro Package

Larry Kollar

kollar@alltel.net

G. Branden Robinson

g.branden.robinson@gmail.com

The *ms* ("manuscript") package is suitable for the composition of letters, memoranda, reports, and books. These *groff* macros feature cover page and table of contents generation, automatically numbered headings, several paragraph styles, a variety of text styling options, footnotes, and multi-column page layouts. *ms* supports the *tbl*, *eqn*, *pic*, and *refer* preprocessors for inclusion of tables, mathematical equations, diagrams, and standardized bibliographic citations. This implementation is mostly compatible with the documented interface and behavior of AT&T Unix Version 7 *ms*. Many extensions from 4.2BSD (Berkeley) and Tenth Edition Research Unix have been recreated.

1. Introduction

The *ms* macros are the oldest surviving package for *roff* systems.[1] While the *man* package was designed for brief reference documents, the *ms* macros are also suitable for longer works intended for printing and possible publication.

In this document, a right arrow (→) is used to indicate a tab character in the input.

1.1. Basic information

ms documents are plain text files; prepare them with your preferred text editor. If you're in a hurry to start, know that *ms* needs one of its macros called at the beginning of a document so that it can initialize. A *macro* is a formatting instruction to *ms*. Put a macro call on a line by itself. Use `.PP` if you want your paragraph's first line to be indented, or `.LP` if you don't.

After that, start typing normally. It is a good practice to start each sentence on a new line, or to put two spaces after sentence-ending punctuation, so that the formatter knows where the sentence boundaries are. You can separate paragraphs with further paragraphing macro calls, or with blank lines, and you can indent with tabs. When you need one of the features mentioned earlier, return to this manual.

Format the document with the *groff*(1) command. *nroff*(1) can be useful for previewing.

```
$ editor radical.ms # vim, emacs, ...
$ nroff -ww -z -ms radical.ms # check for errors
$ nroff -ms radical.ms | less -R
$ groff -T pdf -ms radical.ms > radical.pdf
$ see radical.pdf # or your favorite PDF viewer
```

[1] While manual *pages* are older, early ones used macros supplanted by the *man* package of Seventh Edition Unix (1979). *ms* shipped with Sixth Edition (1975) and was documented by Mike Lesk in a Bell Labs internal memorandum.

Our `radical.ms` document might look like this.

```
.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.
```

ms exposes many aspects of document layout to user control via *groff*'s *registers* and *strings,* which store numbers and text, respectively. Measurements in *groff* are expressed with a suffix called a *scaling unit.*

Scaling unit	Description
`i`	inches (″)
`c`	centimeters
`p`	points (1/72″)
`P`	picas (1/6″)
`v`	"vees"; current vertical spacing
`m`	"ems"; width of an "M" in the current font
`n`	"ens"; one-half em

Set registers with the `nr` request and strings with the `ds` request. *Requests* are like macro calls; they go on lines by themselves and start with the *control character,* a dot (`.`). The difference is that they directly instruct the formatter program, rather than the macro package. We'll discuss a few as applicable. It is wise to specify a scaling unit when setting any register that represents a length, size, or distance.

```
.nr PS 10.5p \" Use 10.5-point type.
.ds FAM P    \" Use Palatino font family.
```

In the foregoing, we see that `\"` begins a comment. This is an example of an *escape sequence,* the other kind of formatting instruction. Escape sequences can appear anywhere. They begin with the escape character (`\`) and are followed by at least one more character. *ms* documents like this one tend to use only a few of *groff*'s many requests and escape sequences; see the *groff*(7) man page for complete lists.

Escape sequence	Description
`\"`	Begin comment; ignore remainder of line.
`\n[`*reg*`]`	Interpolate value of register *reg*.
`\*[`*str*`]`	Interpolate contents of string *str*.
`\*`*s*	abbreviation of `\*[`*s*`]`; the name *s* must be only one character
`\[`*char*`]`	Interpolate glyph of special character named *char*.
`\&`	non-printing, zero-width dummy character
`\~`	Insert an unbreakable space of adjustable width like a normal space.
`|`	Move horizontally by one-sixth em ("thin space").

Prefix any words that start with a dot (`.`) or neutral apostrophe (`'`) with `\&` if they are at the beginning of an input line (or might become that way in editing) to prevent them from being interpreted as macro calls or requests. Suffix `.`, `?`, and `!` with `\&` when needed to cancel end-of-sentence detection.

```
My exposure was \&.5 to \&.6 Sv of neutrons, said Dr.\&
Wallace after the criticality incident.
```

2. General structure of an *ms* document

The *ms* macro package expects a certain amount of structure: a well-formed document contains at least one paragraphing or heading macro call. Longer documents have a structure as follows.

Document type
Calling the `RP` macro at the beginning of your document puts the document description (see below) on a cover page. Otherwise, *ms* places this information (if any) on the first page, followed immediately by the body text. Some document types found in other *ms* implementations are specific to AT&T or Berkeley, and are not supported by *groff ms*.

Format and layout
By setting registers and strings, you can configure your document's typeface, margins, spacing, headers and footers, and footnote arrangement. See section "Document control settings" below.

Document description
A document description consists of any of: a title, one or more authors' names and affiliated institutions, an abstract, and a date or other identifier. See section "Document description macros" below.

Body text
The main matter of your document follows its description (if any). *ms* supports highly structured text consisting of paragraphs interspersed with multi-level headings (chapters, sections, subsections, and so forth) and augmented by lists, footnotes, tables, diagrams, and similar material. See section "Body text" below.

Tables of contents
Macros enable the collection of entries for a table of contents (or index) as the material they discuss appears in the document. You then call a macro to emit the table of contents at the end of your document. The table of contents must necessarily follow the rest of the text since GNU *troff* is a single-pass formatter; it thus cannot determine the page number of a division of the text until it has been set and output. Since *ms* output was designed for the production of hard copy, the traditional procedure was to manually relocate the pages containing the table of contents between the cover page and the body text. Today, page resequencing is more often done in the digital domain. An index works similarly, but because it typically needs to be sorted after collection, its preparation requires separate processing.

3. Document control settings

The document parameters below are presented in the syntax used to interpolate their values. For any unsatisfactory default, define its register, string, or special character before calling any *ms* macro other than `RP`. The "Next" heading indicates that changes to the parameter are effective as of the next new element processed of the listed type. For entries marked *special*, see the discussion in the applicable section.

Type	Parameter	Definition	Next	Default
Margins	`\n[PO]`	Page offset (left margin)	page	`1i (0)`[*]
	`\n[LL]`	Line length	paragraph	`6.5i (65n)`[*]
	`\n[LT]`	Title line length	paragraph	`6.5i (65n)`[*]
	`\n[HM]`	Top (header) margin	page	`1i`
	`\n[FM]`	Bottom (footer) margin	page	`1i`
Titles (headers, footers)	`\*[LH]`	Left header text	header	*empty*
	`\*[CH]`	Center header text	header	`-\n[%]-`
	`\*[RH]`	Right header text	header	*empty*
	`\*[LF]`	Left footer text	footer	*empty*
	`\*[CF]`	Center footer text	footer	*empty*
	`\*[RF]`	Right footer text	footer	*empty*
Text	`\n[PS]`	Type (point) size	paragraph	`10p`
	`\n[VS]`	Vertical spacing	paragraph	`12p`
	`\n[HY]`	Hyphenation mode	paragraph	`6`
	`\*[FAM]`	Font family	paragraph	`T`
Paragraphs	`\n[PI]`	Indentation	paragraph	`5n`
	`\n[PD]`	Paragraph distance (spacing)	paragraph	`0.3v (1v)`[†]
	`\n[QI]`	Quotation indentation	paragraph	`5n`
	`\n[PORPHANS]`	# of initial lines kept	paragraph	`1`
Headings	`\n[PSINCR]`	Type (point) size increment	heading	`1p`
	`\n[GROWPS]`	Size increase depth limit	heading	`0`
	`\n[HORPHANS]`	# of following lines kept	heading	`1`
	`\*[SN-STYLE]`	Numbering style (alias)	heading	`\*[SN-DOT]`
Footnotes	`\n[FI]`	Indentation	footnote	`2n`
	`\n[FF]`	Format	footnote	`0`
	`\n[FPS]`	Type (point) size	footnote	`\n[PS]-2p`
	`\n[FVS]`	Vertical spacing	footnote	`\n[FPS]+2p`
	`\n[FPD]`	Paragraph distance (spacing)	footnote	`\n[PD]/2`
	`\*[FR]`	Line length ratio	*special*	`11/12`
Displays	`\n[DD]`	Display distance (spacing)	*special*	`0.5v (1v)`[†]
	`\n[DI]`	Display indentation	*special*	`0.5i`
Other	`\n[MINGW]`	Minimum gutter width	page	`2n`
	`\n[TC-MARGIN]`	TOC page number margin width	`PX` call	`\w'000'`
	`\[TC-LEADER]`	TOC leader character	`PX` call	`.\h'1m'`

[*] Defaults vary by output device and paper format; the values shown are for typesetters using U.S. letter paper, and then terminals. See section "Paper format" of the *groff*(1) man page.

[†] The `PD` and `DD` registers use the larger value if the vertical motion quantum of the output device is too coarse for the smaller one; usually, this is the case only for output to terminals and emulators thereof.

4. Document description macros

Only the simplest document lacks a title.[2] As its level of sophistication (or complexity) increases, it tends to acquire a date of revision, explicitly identified authors, sponsoring institutions for authors, and, at the rarefied heights, an abstract of its content. By default, *ms* arranges most of the document description (the title, author names and institutions, and abstract, but not the date) at the top of the first page.

Define these data by calling the macros below in the order shown; `.DA` or `.ND` can be called to set the document date (or other identifier) at any time before (a) the abstract, if present, or (b) its information is required in a header or footer. Use of these macros is optional, except that `.TL` is mandatory if any of `.RP`, `.AU`, `.AI`, or `.AB` is called, and `.AE` is mandatory if `.AB` is called.

Macro	**Description**
`.RP` [*option* ...]	Use the "report" (AT&T: "released paper") format for your document, creating a separate cover page. If the optional `no-repeat-info` argument is given, *ms* produces a cover page but does not repeat any of its information subsequently (but see the `DA` macro below regarding the date). Normally, `.RP` sets the page number following the cover page to 1. Specifying the optional `no-renumber` argument suppresses this alteration. Optional arguments can occur in any order. `no` is recognized as a synonym of `no-repeat-info` for AT&T compatibility.
`.TL`	Specify the document title. *ms* collects text on input lines following this call into the title until reaching `.AU`, `.AB`, or a heading or paragraphing macro call.
`.AU`	Specify an author's name. *ms* collects text on input lines following this call into the author's name until reaching `.AI`, `.AB`, another `.AU`, or a heading or paragraphing macro call. Call it repeatedly to specify multiple authors.
`.AI`	Specify the preceding author's institution. An `.AU` call is usefully followed by at most one `.AI` call; if there are more, the last `.AI` call controls. *ms* collects text on input lines following this call into the author's institution until reaching `.AU`, `.AB`, or a heading or paragraphing macro call.
`.DA` [*x* ...]	Typeset the current date, or any arguments *x*, in the center footer, and, if `.RP` is also called, left-aligned at the end of the document description on the cover page.
`.ND` [*x* ...]	Typeset the current date, or any arguments *x*, if `.RP` is also called, left-aligned at the end of the document description on the cover page. This is *groff ms*'s default.
`.AB` [`no`]	Begin the abstract. *ms* collects text on input lines following this call into the abstract until reaching an `.AE` call. By default, *ms* places the word "ABSTRACT" centered and in italics above the text of the abstract. The optional argument `no` suppresses this heading.
`.AE`	End the abstract.

[2] Distinguish a document title from "titles", which are what *roff* systems call headers and footers collectively.

An example document description, using a cover page, follows.

```
.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 more in line with user demand.
.AE
...the rest of the paper...
```

5. Body text

A variety of macros, registers, and strings can be used to structure and style the body of your document. They organize your text into paragraphs, headings, footnotes, and inclusions of material such as tables and figures.

5.1. Text settings

The `FAM` string, a GNU extension, sets the font family for body text; the default is "`T`". The `PS` and `VS` registers set the type size and vertical spacing (distance between text baselines), respectively. The font family and type size are ignored on terminal devices. Setting these parameters before the first call of a heading, paragraphing, or (non-date) document description macro also applies them to headers, footers, and (for `FAM`) footnotes.

Which font families are available depends on the output device; as a convention, `T` selects a serif family ("Times"), `H` a sans-serif family ("Helvetica"), and `C` a monospaced family ("Courier"). The man page for the output driver documents its font repertoire. Consult the *groff*(1) man page for lists of available output devices and their drivers.

The `HY` register defines the automatic hyphenation mode used with the `hy` request. Setting `\n[HY]` to `0` is equivalent to using the `nh` request. This is a Tenth Edition Research Unix extension.

5.2. Typographical symbols

ms provides a few strings to obtain typographical symbols not easily entered with the keyboard. These and many others are available as special character escape sequences—see the *groff_char*(7) man page.

String	Description
`\*[-]`	Interpolate an em dash.
`\*[Q]` `\*[U]`	Interpolate typographer's quotation marks where available, and neutral double quotes otherwise. `\*[Q]` is the left quote and `\*[U]` the right.

5.3. Paragraphs

Paragraphing macros *break*, or terminate, any pending output line so that a new paragraph can begin. Several paragraph types are available, differing in how indentation applies to them: to left, right, or both margins; to the first output line of the paragraph, all output lines, or all but the first. All paragraphing macro calls cause the insertion of vertical space in the amount stored in the `PD` register, except at page or column breaks. Alternatively, a blank input line breaks the output line and vertically spaces by one vee.

The `PORPHANS` register defines the minimum number of initial lines of any paragraph that must be kept together to avoid isolated lines at the bottom of a page. If a new paragraph is started close to the bottom of a page, and there is insufficient space to accommodate `\n[PORPHANS]` lines before an automatic page break, then a page break is forced before the start of the paragraph. This is a GNU extension.

Macro	Description
`.LP`	Set a paragraph without any (additional) indentation.
`.PP`	Set a paragraph with a first-line left indentation of `\n[PI]`.
`.IP` [*marker* [*width*]]	Set a paragraph with a left indentation. The optional *marker* is not indented and is empty by default. It has several applications; see subsection "Lists" below. *width* overrides the indentation amount in `\n[PI]`; its default unit is "`n`". Once specified, *width* applies to further `.IP` calls until specified again or a heading or different paragraphing macro is called.

Macro	**Description**
`.QP`	Set a paragraph indented from both left and right margins by `\n[QI]`.
`.QS` `.QE`	Begin (`QS`) and end (`QE`) a region where each paragraph is indented from both margins by `\n[QI]`. The text between `.QS` and `.QE` can be structured further by use of other paragraphing macros.
`.XP`	Set an "exdented" paragraph—one with a left indentation of `\n[PI]` on every line *except* the first (also known as a hanging indent). This is a Berkeley extension.

The following example illustrates the use of paragraphing macros.

```
.NH 2
Cases used in the 2001 study
.LP
Two software releases were considered for this report.
.PP
The first is commercial software;
the second is free.
.IP \[bu]
Microsoft Word for Windows,
starting with version 1.0 through the current version
(Word 2000).
.IP \[bu]
GNU Emacs,
from its first appearance as a standalone editor through the
current version (v20).
See [Bloggs 2002] for details.
.QP
Franklin's Law applied to software:
software expands to outgrow both RAM and disk space over time.
.SH
Bibliography
.XP
Bloggs, Joseph R.,
.I "Everyone's a Critic" ,
Underground Press, March 2002.
A definitive work that answers all questions and criticisms
about the quality and usability of free software.
```

5.4. 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, type size as the body text. Headings are available with and without automatic numbering. Text on input lines following the macro call becomes the heading's title. Call a paragraphing macro to end the heading text and start the section's content.

Macro	Description
`.NH [`*depth*`]`	Set an automatically numbered heading. *ms* produces a numbered heading in the form *a*.*b*.*c*..., to any level desired, with the numbering of each depth increasing automatically and being reset to zero when a more significant depth increases. "1" is the most significant or coarsest division of the document. Only non-zero values are output. If *depth* is omitted, it is taken to be `1`. If you specify *depth* such that an ascending gap occurs relative to the previous `NH` call—that is, you "skip a depth", as by "`.NH 1`" and then "`.NH 3`"—*groff ms* emits a warning on the standard error stream.
`.NH S` *heading-depth-index* ...	Alternatively, you can give `NH` a first argument of `S`, followed by integers to number the heading depths explicitly. Further automatic numbering, if used, resumes using the specified indices as their predecessors. This feature is a Berkeley extension.

An example may be illustrative.

Input	Result
`.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 `.NH` is called, the assigned number is made available in the strings `SN-DOT` (as it appears in a printed heading with default formatting, followed by a terminating period) and `SN-NO-DOT` (with the terminating period omitted). These are GNU extensions.

You can control the style used to print numbered 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 numbers appearing in numbered headings, you may define the alias as follows.

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

Any such change in numbering style becomes effective from the next use of `.NH` following redefinition of the alias for `\*[SN-STYLE]`. The formatted number of the current heading is available in `\*[SN]` (a feature first documented by Berkeley), which facilitates its inclusion in, for example, table captions, equation labels, and `.XS`/`.XA`/`.XE` table of contents entries.

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

The `PSINCR` register defines an increment in type size to be applied to a heading at a lesser depth than that specified in `\n[GROWPS]`. The value of `\n[PSINCR]` should be specified in points with the `p` scaling unit and may include a fractional component; for example,

```
.nr PSINCR 1.5p
```

sets a type size increment of 1.5 points.

The `GROWPS` register defines the heading depth above which the type size increment set by `\n[PSINCR]` becomes effective. For each heading depth less than the value of `\n[GROWPS]`, the type size is increased by `\n[PSINCR]`. Setting `\n[GROWPS]` to a value less than 2 disables the incremental heading size feature.

In other words, if the value of `GROWPS` register is greater than the *depth* argument to a `.NH` or `.SH` call, the type size of a heading produced by these macros increases by `\n[PSINCR]` units over `\n[PS]` multiplied by the difference of `\n[GROWPS]` and *depth*. 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 headings will remain in the 10-point text specified by the `PS` register. “Machairodontinae” is printed at 11.5 points, since it corresponds to heading depth 2.

The `\n[HORPHANS]` register operates in conjunction with the `NH` and `SH` macros to inhibit the printing of isolated 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 call or *tbl*, *pic*, or *eqn* region (see subsequent sections) between the heading and the subsequent paragraph suppresses this grouping.

5.5. Typeface and decoration

The *ms* macros provide a variety of ways to style text. Attend closely to the ordering of arguments labeled *pre* and *post,* which is not intuitive. Support for *pre* arguments is a GNU extension.[3]

Macro	Description
`.B` [*text* [*post* [*pre*]]]	Style *text* in **bold**, followed by *post* in the previous font style without intervening space, and preceded by *pre* similarly. Without arguments, *ms* styles subsequent text in bold until the next paragraphing, heading, or no-argument typeface macro call.
`.R` [*text* [*post* [*pre*]]]	As `.B`, but use the roman style (upright text of normal weight) instead of bold. Argument recognition is a GNU extension.
`.I` [*text* [*post* [*pre*]]]	As `.B`, but use an *italic* or oblique style instead of bold.
`.BI` [*text* [*post* [*pre*]]]	As `.B`, but use a ***bold italic*** or bold oblique style instead of upright bold. This is a Tenth Edition Research Unix extension.
`.CW` [*text* [*post* [*pre*]]]	As `.B`, but use a `constant-width` (monospaced) roman typeface instead of bold. This is a Tenth Edition Research Unix extension.
`.BX` [*text*]	Typeset *text* and draw a box around it. On terminal devices, reverse video is used instead (see the implementation note below). If you want *text* to contain space, use unbreakable space or horizontal motion escape sequences (`\~`, `\`*space*`, \^, |, \0, or \h).`
`.UL` [*text* [*post*]]	Typeset *text* with an underline. *post,* if present, is set after *text* with no intervening space.
`.LG`	Set subsequent text in larger type (2 points larger than the current size) until the next type size, paragraphing, or heading macro call. You can specify this macro multiple times to enlarge the type size as needed.
`.SM`	Set subsequent text in smaller type (2 points smaller than the current size) until the next type size, paragraphing, or heading macro call. You can specify this macro multiple times to reduce the type size as needed.
`.NL`	Set subsequent text at the normal type size (`\n[PS]`).

pre and *post* arguments are typically used to simplify the attachment of punctuation to styled words. When *pre* is used, a hyphenation control escape sequence `\%` that would ordinarily start *text* must start *pre* instead to have the desired effect.

Input	Result
`The CS course's students found one C language keyword .CW static ) \%( most troublesome.`	The CS course’s students found one C language keyword (`static`) most troublesome.

[3] This idiosyncrasy arose through feature accretion; for example, the `B` macro in Version 6 Unix *ms* (1975) accepted only one argument, the text to be set in boldface. By Version 7 (1979) it recognized a second argument; in 1990, *groff ms* added a “pre” argument, placing it third to avoid breaking support for older documents.

You can use the output line continuation escape sequence `\c` to achieve the same result. It is also portable to some older *ms* implementations.

Input	Result

`The CS course's`
`students found one C`
`language keyword`
`\%(\c`
`.CW static )`
`most troublesome.`
	The CS course's students found one C language keyword (`static`) most troublesome.

Rather than calling the `CW` macro, in *groff ms* you might prefer to change the font family to Courier by setting `\*[FAM]` to "C". You can then use all four style macros above, returning to the default family (Times) with "`.ds FAM T`". Because changes to `\*[FAM]` take effect only at the next paragraph, this document uses `.CW` to "inline" a change to the font family, marking syntactical elements of *ms* and *groff*.

groff ms also offers strings to begin and end super- and subscripting. These are GNU extensions.

String	Description
`\*{`	Begin superscripting.
`\*}`	End superscripting.
`\*<`	Begin subscripting.
`\*>`	End subscripting.

Implementation note: In `nroff` mode, the `BX` macro "boxes" its argument by bracketing it with *groff* extension escape sequences to set the foreground color to black and the background to white and then reset them to their previous values; the terminal output driver, *grotty*(1), converts these to ISO 6429 color escape sequences, which may be ignored or mishandled by some terminals, or may be disabled by *grotty*'s `-c` option. Further, if the terminal is set up to use these colors in those roles already, `.BX` will cause no visible effect in the output. Surmounting these challenges would require adding features to *grotty*, for instance to provide a mechanism to request ISO 6429's "standout" mode (often supported on monochrome terminals), or to replace the presumed support of the terminal for ISO 6429 escape sequences with the use of a library that can query the capabilities of the terminal and adapt the output sent to the device accordingly. (Practically, this likely means adding a dependency on `libtinfo`.) Contact the *groff* development mailing list if you'd like to contribute.

5.6. Lists

The *marker* argument to the `IP` macro can be employed to present a variety of lists; for instance, you can use a bullet glyph (`\[bu]`) for unordered lists, a number (or auto-incrementing register) for numbered lists, or a word or phrase for glossary-style or definition lists. If you set the paragraph indentation register `PI` before calling `IP`, you can later reorder the items in the list without having to ensure that a *width* argument remains affixed to the first call.

Input	Result

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

Input	Result

`.nr step 0 1`
`.nr PI 3n`
`A numbered list:`
`.IP \n+[step].`
`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!

In the enumerated list example, we employed the `nr` request to create a register of our own, `step`. We initialized it to zero and assigned it an auto-increment of 1. Each time we use the escape sequence `\n+[PI]` (note the plus sign), the formatter applies the increment just before interpolating the register's value. Preparing the `PI` register as well enables us to rearrange the list without the tedium of updating macro calls.

In the glossary 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 few workarounds we will illustrate by modifying the example. First, you can use a `br` request to force a break after printing the term or label. Second, you could apply the `\p` escape sequence to force a break. The space following the escape sequence is important; if you omit it, *groff* prints the first word of the paragraph text on the same line as the term or label (if it fits) *then* breaks the line. Finally, you may append a horizontal motion to the marker with the `\h` escape sequence; using the same amount as the indentation will ensure that the marker is too wide for *groff* to treat it as "fitting" on the same line as the paragraph text.

Approach #1	Approach #2	Approach #3

`.IP guns`
`.br`
`Firearms,`
	`.IP guns`
	`\p Firearms,`
		`.IP guns\h'0.4i'`
		`Firearms,`

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

5.7. Indented regions

You may need to indent a region of text while otherwise formatting it normally. Indented regions can be nested; you can change `\n[PI]` before each call to vary the amount of inset.

Macro	Description
`.RS`	Begin a region where headings, paragraphs, and displays are indented (further) by `\n[PI]`.
`.RE`	End the (next) most recent indented region.

This feature enables you to easily line up text under hanging and indented paragraphs. For example, you may wish to structure lists hierarchically.

Input	Result

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

5.8. Keeps, boxed keeps, and displays

On occasion, you may want to *keep* several lines of text, or a region of a document, together on a single page, preventing an automatic page break within certain boundaries. This can cause a page break to occur earlier than it normally would. For example, you may want to keep two paragraphs together, or a paragraph that refers to a table, list, or figure adjacent to the item it discusses. *ms* provides the `KS` and `KE` macros for this purpose.

You can alternatively specify a *floating keep:* if a keep cannot fit on the current page, *ms* holds its contents and allows material following the keep (in the source document) to fill the remainder of the current page. When the page breaks, whether by reaching the end or `bp` request, *ms* puts the floating keep at the beginning of the next page. This is useful for placing large graphics or tables that do not need to appear exactly where they occur in the source document.

Macro	Description
`.KS`	Begin a keep.
`.KF`	Begin a floating keep.
`.KE`	End (floating) keep.

As an alternative to the keep mechanism, the `ne` request forces a page break if there is not at least the amount of vertical space specified in its argument remaining on the page. One application of `ne` is to reserve space on the page for a figure or illustration to be included later.

A *boxed keep* has a frame drawn around it.

Macro	Description
`.B1`	Begin a keep with a box drawn around it.
`.B2`	End boxed keep.

Boxed keep macros cause breaks; if you need to box a word or phrase within a line, see the `BX` macro in section "Typeface and decoration" above. Box lines are drawn as close as possible to the text they enclose so that they are usable within paragraphs. If you wish to place one or more paragraphs in a boxed keep, you may improve their appearance by calling `.B1` after the first paragraphing macro, and by adding a small amount of vertical space before calling `.B2`.

```
.LP
.B1
.I Warning:
Happy Fun Ball may suddenly accelerate to dangerous speeds.
.sp \n[PD]/2 \" space by half the inter-paragraph distance
.B2
```

If you want a boxed keep to float, you will need to enclose the `.B1` and `.B2` calls within a pair of `.KF` and `.KE` calls.

Displays turn off filling; lines of verse or program code are shown with their lines broken as in the source document without requiring `br` requests between lines. Displays can be kept on a single page or allowed to break across pages. The `DS` macro begins a kept display of the layout specified in its first argument; non-kept displays are begun with dedicated macros corresponding to their layout.

Display macro		Description
With keep	**Without keep**	
`.DS L`	`.LD`	Begin left-aligned display.
`.DS [I [`*indent*`]]`	`.ID [`*indent*`]`	Begin display indented by *indent* if given, `\n[DI]` otherwise.
`.DS B`	`.BD`	Begin block display (left-aligned with longest line centered).
`.DS C`	`.CD`	Begin centered display.
`.DS R`	`.RD`	Begin right-aligned display. This is a GNU extension.
`.DE`		End any display.

The distance stored in `\n[DD]` is inserted before and after each pair of display macros; this is a Berkeley extension. In *groff ms*, this distance replaces any adjacent inter-paragraph distance or subsequent spacing prior to a section heading. The `DI` register is a GNU extension; its value is an indentation applied to displays created with `.DS` and `.ID` without arguments, to "`.DS I`" without an indentation argument, and to equations set with "`.EQ I`". Changes to either register take effect at the next display boundary.

5.9. Tables, figures, equations, and references

The *ms* package is often used with the *tbl*, *pic*, *eqn*, and *refer* preprocessors. The `\n[DD]` distance is also applied to regions of the document preprocessed with *tbl*, *pic*, and *eqn*. Mark text meant for preprocessors by enclosing it in pairs of tokens as follows, with nothing between the dot and the macro name. The preprocessors match these tokens only at the start of an input line.

Tag pair	Description
`.TS [H]` `.TE`	Demarcate a table to be processed by the *tbl* preprocessor. The optional `H` argument to `.TS` instructs *ms* to repeat table rows (often column headings) at the top of each new page the table spans, if applicable; calling the `TH` macro marks the end of such rows. The GNU *tbl*(1) man page provides a comprehensive reference to the preprocessor and offers examples of its use.

`.PS`
`.PE`
`.PF`
	`.PS` begins a picture to be processed by the *pic* preprocessor; either of `.PE` or `.PF` ends it, the latter with "flyback" to the vertical position at its top. You can create *pic* input manually or with a program such as *xfig*(1).

`.EQ` [*align* [*label*]] `.EN`	Demarcate an equation to be processed by the *eqn* preprocessor. The equation is centered by default; *align* can be `C`, `L`, or `I` to (explicitly) center, left-align, or indent it by `\n[DI]`, respectively. If specified, *label* is set right-aligned.
`.[` `.]`	Demarcate a bibliographic citation to be processed by the *refer* preprocessor. The GNU *refer*(1) man page provides a comprehensive reference to the preprocessor and the format of its bibliographic database.

When *refer* emits collected references (as might be done on a "Works Cited" page), it interpolates the string `\*[REFERENCES]` as an unnumbered heading (`.SH`).

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

```
.TS H
allbox;
Cb | Cb .
Part→Description
_
.TH
.T&
L | Lx .
GH-1978→Fribulating gonkulator
...the rest of the table follows...
.TE
```

Attempting to place a multi-page table inside a keep can lead to unpleasant results, particularly if the *tbl* `allbox` option is used.

Mathematics can be typeset using the language of the *eqn* preprocessor.

```
.EQ C (\*[SN-NO-DOT]a)
p ~ = ~ q sqrt { 1 + ~ ( x / q sup 2 ) }
.EN
```

This input formats a labelled equation. We used the `SN-NO-DOT` string to base the equation label on the current heading number, giving us more flexibility to reorganize the document.

$$p = q\sqrt{1 + (x/q^2)} \qquad (5.9a)$$

Use *groff* options to run preprocessors on the input: `-e` for *eqn*, `-p` for *pic*, `-R` for *refer*, and `-t` for *tbl*.

5.10. Footnotes

A footnote is typically anchored to a place in the text with a *marker,* which is a small integer,[4] a symbol,[†] or arbitrary user-specified text.

String	Description
`\**`	Place an *automatic number,* an automatically updated numeric footnote marker, in the text. Each time this string is interpolated, the number it produces increments by one. Automatic numbers start at 1. This is a Berkeley extension.

Enclose the footnote text in `FS` and `FE` macro calls to set it at the nearest available "foot", or bottom, of a text column or page.

Macro	Description
`.FS` [*marker*]	Begin a footnote. The `FS-MARK` hook (see below) is called with any supplied *marker* argument, which is then also placed at the beginning of the footnote text. If *marker* is omitted, the next pending automatic number enqueued by interpolation of the `*` string is used, and if none exists, nothing is prefixed.
`.FE`	End footnote text.

You may not desire automatically numbered footnotes in spite of their convenience. You can indicate a footnote with a symbol or other text by specifying its marker at the appropriate place (for example, by using `\[dg]` for the dagger glyph) *and* as an argument to the `FS` macro. Such manual marks should be repeated as arguments to `.FS` or as part of the footnote text to disambiguate their correspondence. You may wish to use `\*{` and `\*}` to superscript the marker at the anchor point, in the footnote text, or both.

groff ms provides a hook macro, `FS-MARK`, for user-determined operations to be performed when the `FS` macro is called. It is passed the same arguments as `FS` itself. An application of `FS-MARK` is anchor placement for a hyperlink reference, so that a footnote can link back to its referential context.[5] By default, this macro has an empty definition. `FS-MARK` is a GNU extension.

The following input was used to produce the first sentence in this section.

```
A footnote is anchored to a place in the text with a
.I marker,
which is a small integer,\**
.FS
like this numeric footnote
.FE
a symbol,\*{\[dg]\*}
.FS \[dg]
like this symbolic footnote
.FE
or arbitrary user-specified text.
```

Footnotes can be safely used within keeps and displays, but you should avoid using automatically numbered footnotes within floating keeps. You can place a second `\**` interpolation between a `\**` and its corresponding `.FS` call as long as each `.FS` call occurs *after* the corresponding `\**` and occurrences of `.FS` are in the same order as corresponding occurrences of `\**`.

---

4 like this numeric footnote

† like this symbolic footnote

5 "Portable Document Format Publishing with GNU Troff", *pdfmark.ms* in the *groff* distribution, uses this technique.

Footnote text is formatted as paragraphs are, using analogous parameters. The registers `FI`, `FPD`, `FPS`, and `FVS` correspond to `PI`, `PD`, `PS`, and `VS`, respectively; `FPD`, `FPS`, and `FVS` are GNU extensions.

The `FF` register controls the formatting of automatically numbered footnote paragraphs, and those for which `.FS` is given a *marker* argument, at the bottom of a column or page as follows.

FF value	**Description**
0	Set an automatic number as a superscript (on typesetter devices) or surrounded by square brackets (on terminals). The footnote paragraph is indented as with `.PP` if there is an `.FS` argument or an automatic number, and as with `.LP` otherwise. This is the default.
1	As `0`, but set the marker as regular text, and follow an automatic number with a period.
2	As `1`, but without indentation (like `.LP`).
3	As `1`, but set the footnote paragraph with the marker hanging (like `.IP`).

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,[6] 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.[7]

`\*[FR]` should be used in preference to `\n[FL]` in contemporary documents. The footnote line length is effectively computed as "*column-width* `*` `\*[FR]`". If an absolute footnote line length is required, recall that arithmetic expressions in the *roff* language are evaluated strictly from left to right, with no operator precedence (parentheses are honored).

```
.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.

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

[7] You could reset it after each call to `.1C`, `.2C`, or `.MC`.

5.11. Language and localization

groff ms provides several strings that you can customize for your own purposes, or redefine to adapt the macro package to languages other than English. It is already localized for Czech, German, French, Italian, and Swedish. Load the desired localization macro package after *ms*; see the *groff_tmac*(5) man page.

```
$ groff -ms -mfr bienvenue.ms
```

The following strings are available.

String	Default
`\*[REFERENCES]`	`References`
`\*[ABSTRACT]`	`\f[I]ABSTRACT\f[]`
`\*[TOC]`	`Table of Contents`
`\*[MONTH1]`	`January`
`\*[MONTH2]`	`February`
`\*[MONTH3]`	`March`
`\*[MONTH4]`	`April`
`\*[MONTH5]`	`May`
`\*[MONTH6]`	`June`
`\*[MONTH7]`	`July`
`\*[MONTH8]`	`August`
`\*[MONTH9]`	`September`
`\*[MONTH10]`	`October`
`\*[MONTH11]`	`November`
`\*[MONTH12]`	`December`

The default for `\*[ABSTRACT]` includes font selection escape sequences to set the word in italics.

5.12. Glyphs for special characters

Some of the special character escape sequences used in this document are listed below. The minus sign glyph can also be accessed by the shorthand `\-`. These and many others are documented in the *groff_char*(7) man page.

Input	Appearance	Description
`\[-]`	−	minus sign
`\[->]`	→	right arrow
`\[aq]`	'	neutral apostrophe
`\[bu]`	•	bullet
`\[dq]`	"	neutral double quote
`\[dg]`	†	dagger
`\[em]`	—	em dash
`\[ha]`	^	circumflex accent (caret, hat)
`\[lq]`	“	left double quotation mark
`\[rq]`	”	right double quotation mark
`\[rs]`	\	reverse solidus (backslash)
`\[sd]`	″	seconds (double prime) mark
`\[ti]`	~	tilde

6. Page layout

ms's default page layout arranges text in a single column with the page number between hyphens centered in a header on each page except the first, and produces no footers. You can customize this arrangement.

6.1. Headers and footers

There are multiple ways to produce headers and footers. One is to define the strings `LH`, `CH`, and `RH` to set the left, center, and right headers, respectively; and `LF`, `CF`, and `RF` to set the left, center, and right footers. This approach suffices for documents that do not distinguish odd- and even-numbered pages.

Another method is to call macros that set headers or footers for odd- or even-numbered pages. Each such macro takes a delimited argument separating the left, center, and right header or footer texts from each other. You can replace the neutral apostrophes (') shown below with any character not appearing in the header or footer text. These macros are Berkeley extensions.

Macro	Description
`.OH` `'`*left*`'`*center*`'`*right*`'`	Set the left, center, and right headers on odd-numbered (recto) pages.
`.OF` `'`*left*`'`*center*`'`*right*`'`	Set the left, center, and right footers on odd-numbered (recto) pages.
`.EH` `'`*left*`'`*center*`'`*right*`'`	Set the left, center, and right headers on even-numbered (verso) pages.
`.EF` `'`*left*`'`*center*`'`*right*`'`	Set the left, center, and right footers on even-numbered (verso) pages.

With either method, a percent sign **%** in header or footer text is replaced by the current page number. By default, *ms* places no header on a page numbered "1" (regardless of its number format).

Macro	Description
`.P1`	Typeset the header even on page 1. To be effective, this macro must be called before the header trap is sprung on any page numbered "1"; in practice, unless your page numbering is unusual, this means that you should call it early, before `.TL` or any heading or paragraphing macro. This is a Berkeley extension.

For even greater flexibility, *ms* is designed to permit the redefinition of the macros that are called when the *groff* traps that ordinarily cause the headers and footers to be output are sprung. `PT` ("page trap") is called by *ms* when the header is to be written, and `BT` ("bottom trap") when the footer is to be. The *groff* page location trap that *ms* sets up to format the header also calls the (normally undefined) `HD` macro after `.PT`; you can define `.HD` if you need additional processing after setting the header (for example, to draw a line below it). The `HD` hook is a Berkeley extension. Any such macros you (re)define must implement any desired specialization for odd-, even-, or first numbered pages.

6.2. Tab stops

Use the `ta` request to set tab stops as needed.

Macro	Description
`.TA`	Reset the tab stops to the *ms* default (every 5 ens). Redefine this macro to create a different set of default tab stops.

6.3. Margins

Control margins using the registers summarized in the "Margins" portion of the table in section "Document control settings" above. There is no setting for the right margin; the combination of page offset `\n[PO]` and line length `\n[LL]` determines it.

6.4. Multiple columns

ms can set text in as many columns as reasonably fit on the page. The following macros force a page break if a multi-column layout is active when they are called. `\n[MINGW]` is the default minimum gutter width; it is a GNU extension. When multiple columns are in use, keeps and the `HORPHANS` and `PORPHANS` registers work with respect to column breaks instead of page breaks.

Macro	Description
`.1C`	Arrange page text in a single column (the default).
`.2C`	Arrange page text in two columns.
`.MC` [*column-width* [*gutter-width*]]	Arrange page text in multiple columns. If you specify no arguments, it is equivalent to the `2C` macro. Otherwise, *column-width* is the width of each column and *gutter-width* is the minimum distance between columns.

6.5. Creating a table of contents

Because *roff* formatters process their input in a single pass, material on page 50, for example, cannot influence what appears on page 1—this poses a challenge for a table of contents at its traditional location in front matter, if you wish to avoid manually maintaining it. *ms* enables the collection of material to be presented in the table of contents as it appears, saving its page number along with it, and then emitting the collected contents on demand toward the end of the document. The table of contents can then be resequenced to its desired location by physically rearranging the pages of a printed document, or as part of post-processing—with a *sed*(1) script to reorder the pages in *troff*'s output, with *pdfjam*(1), or with *gropdf*(1)'s **.pdfswitchtopage** feature, for example.

Define an entry to appear in the table of contents by bracketing its text between calls to the `XS` and `XE` macros. A typical application is to call them immediately after `NH` or `SH` and repeat the heading text within them. The `XA` macro, used within `.XS`/`.XE` pairs, supplements an entry—for instance, when it requires multiple output lines, whether because a heading is too long to fit or because style dictates that page numbers not be repeated. You may wish to indent the text thus wrapped to correspond to its heading depth; this can be done in the entry text by prefixing it with tabs or horizontal motion escape sequences, or by providing a second argument to the `XA` macro. `.XS` and `.XA` automatically associate the page number where they are called with the text following them, but they accept arguments to override this behavior. At the end of the document, call `TC` or `PX` to emit the table of contents; `.TC` resets the page number to **i** (Roman numeral one), and then calls `PX`. All of these macros are Berkeley extensions.

Macro	Description
`.XS` [*page-number*] `.XA` [*page-number* [*indentation*]] `.XE`	Begin, supplement, and end a table of contents entry. Each entry is associated with *page-number* (otherwise the current page number); a *page-number* of `no` prevents a leader and page number from being emitted for that entry. Use of `.XA` within `.XS`/`.XE` is optional; it can be repeated. If *indentation* is present, a supplemental entry is indented by that amount; ens are assumed if no unit is indicated. Text on input lines between `.XS` and `.XE` is stored for later recall by `.PX`.
`.PX` [`no`]	Switch to single-column layout. Unless `no` is specified, center and interpolate `\*[TOC]` in bold and two points larger than the body text. Emit the table of contents entries.
`.TC` [`no`]	Set the page number to 1, the page number format to lowercase Roman numerals, and call `PX` (with a `no` argument, if present).

Here's an example of typical *ms* table of contents preparation and its result. We employ horizontal escape sequences `\h` to indent the entries by sectioning depth.

```
.NH 1
Introduction
.XS
Introduction
.XE
...
.NH 2
Methodology
.XS
\h'2n'Methodology
.XA no
\h'4n'Fassbinder's Approach
.XA no
\h'4n'Kahiu's Approach
.XE
...
.NH 1
Findings
.XS
Findings
.XE
...
.TC
```

Table of Contents

The remaining features in this subsection are GNU extensions. *groff ms* obviates the need to repeat heading text after `XS` calls. Call `XN` and `XH` after `NH` and `SH`, respectively.

Macro	Description
`.XN` *heading-text*	Format *heading-text* and create a corresponding table of contents entry; the indentation is computed from the depth of the preceding `NH` call.
`.XH` *depth heading-text*	As `.XN`, but use *depth* to determine the indentation.

groff ms encourages customization of table of contents entry production.

Macro	Description
`.XN-REPLACEMENT` *heading-text* `.XH-REPLACEMENT` *depth heading-text*	These hook macros implement `.XN` and `.XH`, respectively. They call `XN-INIT` and pass their *heading-text* arguments to `.XH-UPDATE-TOC`.
`.XH-INIT`	This hook macro does nothing by default.
`.XH-UPDATE-TOC` *depth heading-text*	Bracket *heading-text* with `XS` and `XE` calls, indenting it by 2 ens per level of *depth* beyond the first.

```
.NH 1
.XN Introduction
...
.NH 2
.XN Methodology
.XH 3 "Fassbinder's Approach"
.XH 3 "Kahiu's Approach"
...
.NH 1
.XN Findings
...
.TC
```

Table of Contents

Introduction 1
  Methodology 2
    Fassbinder's Approach 2
    Kahiu's Approach 2
Findings 5

To get the section number of the numbered headings into the table of contents entries, we might define `XN-REPLACEMENT` as follows. (We obtain the heading depth from *groff ms*'s internal register `nh*hl`.)

```
.de XN-REPLACEMENT
.XN-INIT
.XH-UPDATE-TOC \\n[nh*hl] \\$@
\&\\*[SN] \\$*
..
```

You can change the style of the leader that bridges each table of contents entry with its page number; define the `TC-LEADER` special character by using the `char` request. A typical leader combines the dot glyph "`.`" with a horizontal motion escape sequence to spread the dots. The width of the page number field is stored in the `TC-MARGIN` register.

7. Differences from AT&T *ms*

The *groff ms* macros are an independent reimplementation, using no AT&T code. Since they take advantage of the extended features of *groff*, they cannot be used with AT&T *troff*. *groff ms* supports features described above as Berkeley and Tenth Edition Research Unix extensions, and adds several of its own.

The internals of *groff ms* differ from those of AT&T *ms*. Documents that depend upon implementation details of AT&T *ms* may not format properly with *groff ms*. Such details include macros whose function was not documented in the AT&T *ms* manual [Lesk 1978].

The error-handling policy of *groff ms* is to detect and report errors, rather than to ignore them silently.

Tenth Edition Research Unix supported **P1/ P2** macros to bracket code examples; *groff ms* does not.

groff ms does not work in GNU *troff*'s AT&T compatibility mode. If loaded when that mode is enabled, it aborts processing with a diagnostic message.

Multiple line spacing is not supported. Use a larger vertical spacing instead.

groff ms uses the same header and footer defaults in both `nroff` and `troff` modes as AT&T *ms* does in `troff` mode; AT&T's default in `nroff` mode is to put the date, in U.S. traditional format (e.g., "January 1, 2021"), in the center footer (the `CF` string).

Many *groff ms* macros, including those for paragraphs, headings, and displays, cause a reset of formatting parameters, and may change the indentation; they do so not by incrementing or decrementing it, but by setting it absolutely. This can cause problems for documents that define additional macros of their own that try to manipulate indentation. Use `.RS` and `.RE` instead of the `in` request.

AT&T *ms* interpreted the values of the registers `PS` and `VS` in points, and did not support the use of scaling units with them. *groff ms* interprets values of the registers `PS`, `VS`, `FPS`, and `FVS` equal to or larger than 1,000 (one thousand) as decimal fractions multiplied by 1,000.[8] This threshold makes use of a scaling unit with these parameters practical for high-resolution devices while preserving backward compatibility. It also permits expression of non-integral type sizes. For example, "`groff -rPS=10.5p`" at the shell prompt is equivalent to placing "`.nr PS 10.5p`" at the beginning of the document.

AT&T *ms*'s `AU` macro supported arguments used with some document types; *groff ms* does not.

Right-aligned displays are available. The AT&T *ms* manual observes that "it is tempting to assume that `.DS R` will right adjust lines, but it doesn't work". In *groff ms*, it does.

To make *groff ms* use the default page offset (which also specifies the left margin), the **PO** register must stay undefined until the first *ms* macro is called. This implies that **\n[PO]** should not be used early in the document, unless it is changed also: accessing an undefined register automatically defines it.

groff ms supports the `PN` register, but it is not necessary; you can access the page number via the usual `%` register and invoke the `af` request to assign a different format to it if desired.[9]

The AT&T *ms* manual documents registers `CW` and `GW` as setting the default column width and "intercolumn gap", respectively, and which applied when `.MC` was called with fewer than two arguments. *groff ms* instead treats `.MC` without arguments as synonymous with `.2C`; there is thus no occasion for a default column width register. Further, the `MINGW` register and the second argument to `.MC` specify a *minimum* space between columns, not the fixed gutter width of AT&T *ms*.

The AT&T *ms* manual did not document the `QI` register; Berkeley and *groff ms* do.

The register `GS` is set to 1 by the *groff ms* macros, but is not used by the AT&T *ms* package. Documents that need to determine whether they are being formatted with *groff ms* or another implementation should test this register.

---

[8] Register values are converted to and stored as basic units. See "Measurements" in the *groff* Texinfo manual or in *groff*(7).

[9] If you redefine the *ms* `PT` macro and desire special treatment of certain page numbers (like "`1`"), you may need to handle a non-Arabic page number format, as *groff ms*'s `.PT` does; see the macro package source. *groff ms* aliases the `PN` register to `%`.

7.1. Unix Version 7 *ms* macros not implemented by *groff ms*

Several macros described in the Unix Version 7 *ms* documentation are unimplemented by *groff ms* because they are specific to the requirements of documents produced internally by Bell Laboratories, some of which also require a glyph for the Bell System logo that *groff* does not support. These macros implemented several document type formats (`EG`, `IM`, `MF`, `MR`, `TM`, `TR`), were meaningful only in conjunction with the use of certain document types (`AT`, `CS`, `CT`, `OK`, `SG`), stored the postal addresses of Bell Labs sites (`HO`, `IH`, `MH`, `PY`, `WH`), or lacked a stable definition over time (`UX`). To compatibly render historical *ms* documents using these macros, we advise your documents to invoke the `rm` request to remove any such macros it uses and then define replacements with an authentically typeset original at hand.[10] For informal purposes, a simple definition of `UX` should maintain the readability of the document's substance.

```
.rm UX
.ds UX Unix\"
```

[10] The removal beforehand is necessary because *groff ms* aliases these macros to a diagnostic macro, and you want to redefine the aliased name, not its target.

8. Legacy features

groff ms retains some legacy features solely to support formatting of historical documents; contemporary ones should not use them because they can render poorly. See *groff_char*(7) instead.

8.1. AT&T *ms* accent mark strings

AT&T *ms* defined accent mark strings as follows.

String	Description
`\*[']`	Apply acute accent to subsequent glyph.
`` \*[`] ``	Apply grave accent to subsequent glyph.
`\*[:]`	Apply dieresis (umlaut) to subsequent glyph.
`\*[^]`	Apply circumflex accent to subsequent glyph.
`\*[~]`	Apply tilde accent to subsequent glyph.
`\*[C]`	Apply caron to subsequent glyph.
`\*[,]`	Apply cedilla to subsequent glyph.

8.2. Berkeley *ms* accent mark and glyph strings

Berkeley *ms* offered an `AM` macro; calling it redefined the AT&T accent mark strings (except for `\*C`), applied them to the *preceding* glyph, and defined additional strings, some for spacing glyphs.

Macro	Description
`.AM`	Enable alternative accent mark and glyph-producing strings.

String	Description
`\*[']`	Apply acute accent to preceding glyph.
`` \*[`] ``	Apply grave accent to preceding glyph.
`\*[:]`	Apply dieresis (umlaut) to preceding glyph.
`\*[^]`	Apply circumflex accent to preceding glyph.
`\*[~]`	Apply tilde accent to preceding glyph.
`\*[,]`	Apply cedilla to preceding glyph.
`\*[/]`	Apply stroke (slash) to preceding glyph.
`\*[v]`	Apply caron to preceding glyph.
`\*[_]`	Apply macron to preceding glyph.
`\*[.]`	Apply underdot to preceding glyph.
`\*[o]`	Apply ring accent to preceding glyph.
`\*[?]`	Interpolate inverted question mark.
`\*[!]`	Interpolate inverted exclamation mark.
`\*[8]`	Interpolate small letter sharp s.
`\*[q]`	Interpolate small letter o with hook accent (ogonek).
`\*[3]`	Interpolate small letter yogh.
`\*[d-]`	Interpolate small letter eth.
`\*[D-]`	Interpolate capital letter eth.
`\*[th]`	Interpolate small letter thorn.
`\*[TH]`	Interpolate capital letter thorn.
`\*[ae]`	Interpolate small ae ligature.
`\*[AE]`	Interpolate capital ae ligature.
`\*[oe]`	Interpolate small oe ligature.
`\*[OE]`	Interpolate capital oe ligature.

9. Further reading

"Typing Documents on the Unix System: Using the -ms Macros with Troff and Nroff"; M. E. Lesk; November 13, 1978. This document describes the *ms* supplied with AT&T Unix Version 7.

"A Revised Version of -ms"; Bill Tuthill; August 1983. The 4.2BSD release featured several extensions to *ms*, most of which are recreated in *groff ms*. (The exceptions are the `TM` and `CT` macros for setting a doctoral thesis in the format required by the contemporaneous degree-granting authorities of the University of California at Berkeley.)

"Using PDF boxes with *groff* and the *ms* macros"; Deri James; March 2021. `BOXSTART` and `BOXSTOP` macros are available via the *sboxes* extension package, enabling colored, bordered boxes when the `pdf` output device is used. This document is distributed with *groff* as the file *msboxes.pdf*.

This manual was composed using *groff ms*; the curious may consult its source in the file *ms.ms* to see how its formatting was achieved.

Table of Contents