HEWLETT PACKARD

Text Formatters
HP-UX Concepts and Tutorials

Text Formatters
HP-UX Concepts and Tutorials

HP Part Number 97089-90032

Hewlett-Packard Company
3404 East Harmony Road, Fort Collins, Colorado 80525

Printing History

New editions of this manual will incorporate all material updated since the previous edition. Update packages may be issued between editions and contain replacement and additional pages to be merged into the manual by the user. Each updated page will be indicated by a revision date at the bottom of the page. A vertical bar in the margin indicates the changes on each page. Note that pages which are rearranged due to changes on a previous page are not considered revised.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

August 1986...Edition 1

Table of Contents

Nroff/Troff Text Processors

Introduction

TROFF is **not** *supported on Series 9000 workstations at this time.*

nroff and **troff** are text processors under the HP-UX Operating System. They accept lines of text, interspersed with lines of format control information and format the text into a printable, paginated documentation having a user-designed style. They offer unusual freedom in document styling, including:

- arbitrary style headers and footers.
- arbitrary style footnotes.
- multiple automatic sequence numbering for paragraphs.
- a family of automatic overstriking, bracket construction and line drawing functions.

These text processors are highly compatible with each other and it is almost always possible to prepare input acceptable to both. Conditional input is provided that enables the user to embed input expressly destined for either program. They can prepare output directly for a variety of HP terminals and are capable of utilizing the full resolution of each terminal.

Usage

The general form of invoking **nroff** at HP-UX command level is:

```
nroff [options] file_name_list
```

where options represents any number of option arguments and `file_name_list` represents the list of files containing the document to be formatted. An argument consisting of a single minus (-) is taken to be a file name corresponding to the standard input. If no file names are given, input is taken from the standard input. The options, which may appear in any order so long as they appear before the `file_name_list` are:

option	Effect
-olist	Print only pages whose page numbers appear in *list*, which consists of comma-separated numbers and number ranges. A number range has the form *N-M* and means pages *N* through M inclusive. An initial *-N* means from the beginning (page 1) to page *N* and a final *N-* means use page numbers from *N*.
-nN	Number first generated page *N*.
-sN	Stop every *N* pages. **nroff** will halt prior to every *N* pages (default N = 1) to allow paper loading or changing, and will resume upon receipt of a new-line character.
-nname	Attaches the macro file */usr/lib/tmac.name* to the input `file_name_list`.
-cname	Same as -nname, but uses a compacted form of /usr/lib/tmac.name for the sake of efficiency.
-raN	Register a (register name limited to one character) is set to the value *N*.
-i	Read standard input after the *file_name_list* files are exhausted.
-q	Invoke the simultaneous input-output mode of the **rd** request.
	nroff Only
-Tname	Specifies the name of the output terminal type.
-e	Produce equally spaced words in adjusted lines, using full terminal resolution.

Each option is invoked as a separate argument, for example:

```
nroff -o4,8-10  -mabc file1 file2
```

requests formatting of pages 4, 8, 9, and 10 of a document contained in the files `file1` and `file2` and invoking the macropackage `abc`.

A reverse-line postprocessor `col(1)` is available for multiple-column output on terminals without reverse-line ability.

Summary and Index

General Explanation

The following is a list of supplied formatting constructs. These "dot constructs" must begin in the far left margin of a line and must be the only entry on that line. The **Notes#** refered to are:

Note	Meaning
1	No effect in **nroff**.
2	Values separated by ";" are for **nroff** and **troff** respectively.
3	The use of "'" as control character (instead of ".") suppresses the break function.
4	Request normally causes a break.
5	Mode or relevant parameters associated with current diversion level.
6	Relevant parameters are a part of the current environment.
7	Must stay in effect until logical output.
8	Mode must be still or again in effect at the time of physical output.
9	Default scale indicator; it not specified, scale indicators are ignored.

Font Control

Request Form	Initial Value	If No Argument	Notes#	Explanation
.ft F	Roman	previous	5	Change to font F = x. Also \fx.

Page Control

Request Form	Initial Value	If No Argument	Notes#	Explanation
.pl ±N	11 in	11 in	9	Page length.
.bp ±N	N = 1	-	3,9	Eject current page; next page number N.
.pn ±N	N = 1	ignored	-	Next page number N.
.po ±N	0; 26/27 in	previous	9	Page offset
.ne N	-	N = 1 V	5,9	Need N vertical space (V = vertical spacing).
.mk R	none	internal	5	Mark current vertical place in register R.
.rt ±N	none	internal	5,9	Return (upward only) to marked vetical place.

Text Filling, Adjusting and Centering

Request Form	Initial Value	If No Argument	Notes#	Explanation
.br	-	-	4	Break.
.fi	fill	-	4,6	Fill output lines.
.nf	fill	-	4,6	No filling or adjusting of output lines.
.ad c	adj,both	adjust	6	Adjust output lines with mode c.
.na	adjust	-	6	No output line adjusting.
.ce N	off	N = 1	4,6	Center following N input text lines.

Vertical Spacing

Request Form	Initial Value	If No Argument	Notes#	Explanation
.vs N	1/6in	previous	6,9	Vertical base line spacing (V).
.ls N	N = 1	previous	6	Output N-1 Vs after each text output line.
.sp N	-	N = 1V	6,9	Save vertical distance N in either direction.
.sv N	-	N = 1V	9	Save vertical distance N.
.os	-	-	-	Output saved vertical distance.
.ns	space	-	5	Turn no-space mode on.
.rs	-	-	5	Restore spacing; turn no-space mode off.

Line Length and Indenting

Request Form	Initial Value	If No Argument	Notes#	Explanation
.ll ±N	6.5 in	previous	6,9	Line length.
.in ±N	N = 0	previous	4,6,9	Indent.
.ti ±N	-	ignored	4,6,9	Temporary indent.

Macros, Strings, Diversion and Position Traps

Request Form	Initial Value	If No Argument	Notes#	Explanation
.de xx yy	-	.yy = ..	-	Define or redefine macro xx; end at call of yy.
.am xx yy	-	.yy = ..	-	Append to macro.
.ds xx string	-	ignored	-	Define a string xx containing string.
.as xx string	-	ignored	-	Append string to string xx.
.rm xx	-	ignored	-	Remove request, macro or string.
.rn xx yy	-	ignored	-	Rename request, macro or string xx to yy.
.di xx	-	end	5	Divert output to macro xx.
.da xx	-	end	5	Divert and append to xx.
.wh N xx	-	-	9	Set location trap; negative is w.r.t. page bottom.
.ch xx N	-	-	9	Change trap location.
.dt N xx	-	off	5,9	Set a diversion trap.
.it N xx	-	off	6	Set an input-line count trap.
.em xx	none	none	-	End macro is xx.

Number Registers

Request Form	Initial Value	If No Argument	Notes#	Explanation
.nr R ±N M		-	9	Define and set number register R; auto-increment by M.
.ar R c	arabic	-	-	Assign format to register R (c = 1,i,I,a,A).
.rr R	-	-	-	Remove register R.

Tabs, Leaders and Fields

Request Form	Initial Value	If No Argument	Notes#	Explanation
.ta Nt ...	0.8; 0.6in	none	6,9	Tab settings; left type, unless t = R(right), C(centered).
.tc c	none	none	6	Tab repetition character.
.lc c	.	none	6	Leader repetition character.
.fc a b	off	off	-	Set field delimiter a and pad character b.

I/O Conventions and Character Translations

Request Form	Initial Value	If No Argument	Notes#	Explanation
.ec c	\	\	-	Set excape character.
.eo	on	-	-	Turn off escape character mechanism.
.lg N	-;on	on	-	Ligature mode on if N>0.
.ul N	off	N = 1	6	Underline (italicize in **troff**) N input lines.
.cu N	off	N = 1	6	Continuous underline in **nroff**; like ul in **troff**.
.uf F	Italic	Italic	-	Underline font set to F (to be switched to by ul).
.cc c	.	.	6	Set control character to c.
.c2 c	'	'	6	Set nobreak control character to c.
.tr abcd...	none	-	7	Translate a to b, etc. on output.

Hyphenation

Request Form	Initial Value	If No Argument	Notes#	Explanation
.nh	hyphenate	-	6	No hyphenation
.hy N	hyphenate	hyphenate	6	Hyphenate; N = mode.
.hc c	\%	\%	6	Hyphenation indicator character c.
.hw word1...		ignored	-	Exception words.

Three Part Titles

Request Form	Initial Value	If No Argument	Notes#	Explanation
.tl 'left'center'right'		-	-	Three part title.
.pc c	%	off	-	Page number character.
.lt ±N	6.5in	previous	6,9	Length of title.

Output Line Numbering

Request Form	Initial Value	If No Argument	Notes#	Explanation
.nm ±N M S I		off	6	Number mode on or off, set parameters.
.nn N	-	previous	6	Do not number next N lines.

Conditional Acceptance of Input

Request Form	Initial Value	If No Argument	Notes#	Explanation
.if c anything		-	-	If condition c true, accept anything as input, for multi-line ust\{anything\}.
.if !2c anything		-	-	If condition c false, accept anything as input.
.if N anything		-	9	If expression N>0, accept anything.
.if !N anything		-	9	If expression N≤0, accept
.if 'string1'string2' anything			-	If string1 identical to string2, accept anything.
.if !'string1'string2' anything			-	If string1 not identical to string2, accept anything.
.ie c anything		-	9	If portion of if-else; all above corms (like if).
.el anything		-	-	Else portion of if-else.

Environmental Switching

Request Form	Initial Value	If No Argument	Notes#	Explanation
E.ev N	N = 0	previous	-	Environment switched (push down).

Insertions From Standard Input

Request Form	Initial Value	If No Argument	Notes#	Explanation
.rd prompt	-	prompt = BEL	-	Read insertion.
.ex	-	-	-	Exit from **nroff**.

Input/Output File Switching

Request Form	Initial Value	If No Argument	Notes#	Explanation
.so filename		-	-	Switch source file (push down).
.nx filename		end-of-file	-	Next file.
.pi program		-	-	Pipe output to program (**nroff** only).

Miscellaneous

Request Form	Initial Value	If No Argument	Notes#	Explanation
.mc c N	-	off	6,9	Set margin character c and separation N.
.tm string	-	newline	-	Print string on terminal, (HP-UX standard message output)
.ig yy	-	.yy=..	-	Ignore till call of yy.
.pm t	-	all	-	Print macro names and sizes; if t present, prent only total of sizes.
.fl	-	-	4	Flush output buffer.

Escape Sequences

Escape Sequence	Meaning
\\	\ (to prevent or delay the interpretation of \).
\e	Printable version of the current escape character.
\'	(acute accent); equivalent to \(a.a
\`	(grave accent); equivalent to \g.a
\-	Minus sign in the current font.
\.	Period (dot) (see de).
\(space)	Unpaddable space-size space character.
\0	Digit width space.
\	Zero width in **nroff**.
\^	zero width in **nroff**.
\&	Non-printing, zero width charactr.r
\!	Transparent line indicator.
\"	Beginning of comment.
\$N	Interpolate argument $1 \leq N \leq 9$.
\%	Default optional hyphenation character.
\(xx	Character named xx.
*x, *(xx	Interpolate string x or xx.
\a]	Non-interpreted leader character.
\b 'abc...'	Bracket building function.
\c	Interrupt text processing.
\fx	Change to font named x.
\h'N '	Local horizontal motion; move right N (negative left)

\kx	Mark horizontal input place inregister x
\l'Nc'	Horizontal line drawing function (optionally with c)
\L'Nc'	Vertical line drawing function (optionally with c)
\nx, \n(xx	Interpolate number register x or xx
\o'abc...'	Overstrike characters a, b, c, ...
\p	Break and spread output line
\r	Reverse line in **nroff**.
\t	Non-interpreted horizontal tab.
\u	1/2 line in **nroff**.
\v'N'	Local vertical motion; move down N (negative up).
\w'string'	Interpolate width of string.
\x'N'	Extra line-space function (negative before, positive afte).)
\zc	Print c with zero width (without spacing).
\{	Begin conditional input.
\}	End conditional input.
\(new-line)	Concealed (ignored) new-line.
\X	X, any character not listed aboe.e

The escape sequences \\, \., \'', \$, *, \a, \n, \t,and \(newline) are in copy mode.

Predefined General Number Registers

Register Name	Description
%	Current page number.
ct	Character type (set by width function).
dl	Width (maximum) of last completed diversion.
dn	Height (verticalsize) of last completed diversion.
dw	Current day of the week (1-7).
dy	Current day of the month (1-31).
hp	Current horizontal place on input line.
ln	Output line number.
mo	Current month (1-12).
nl	Vertical position of last printed text base-line.
sb	Depth of string below base in e (generated by width function).
st	Height of string above base line (generated by width function).
yr	Last two digits of curret tyear.

Predefined Read-Only Number Registers

Register Name	Description
.$	Number of arguments available at the current macro level.
.A	Set to 1 in , if -a option used; always 1 in .
.H	Available horizontal resolution in basic units.
.T	Set to 1 in , if -T option used; always 0 in .
.V	Available vertical resolution in basic units.
.a	Post-line extra line-space most recently utilized using \x'N'.
.c	Number of lines read from current input file.
.d	Current vertical place in current diversion; equal to nl, if no diversion.
.h	Text base-line high-water mark on current page or diversion.
.i	Current indent.
.l	Current line length.
.n	Length of text portion on previous output line.
.o	Current page offset.
.p	Current page length.
.s	Current point size.
.t	Distance to the next trap.
.u	Equal to 1 in fill mod and 0 in nofill mode.
.v	Current vertical line spacing.
.w	Width of previous character.
.x	Reserved version-dependent register.
.y	Reserved version-dependent register.
.z	Name of current diversion.

Nroff/Troff Reference

The Basics

Form of Input

Input consists of *text lines*, which are destined to be printed, interspersed with *control lines*, which set parameters or otherwise control subsequent processing. Control lines begin with a control character; normally a period (.) or single-quote ('), followed by a one or two character name that specifies a basic request or the substitution of a user-defined macro in place of the control line. The single-quote control character suppresses the break function. The break function forces output of a partially filled line and is caused by certain requests. The control character may be separated from these (spaces and/or tabs) for esthetic reasons. Names must be followed by either a space or new-line character. Control lines with unrecognized names are ignored.

Various special functions may be introduced anywhere in the input by means of an escape character, normally a backslash (\). For example, the function \nR causes the interpolation of the contents of the number register R in place of the function; here R is either a single character name as in \nx, or left-parenthesis-introduced, two-character name as in \n(xx.

Formatter Resolution

nroff internally uses 240 units/inch, corresponding to the least common multiple of the horizontal and vertical resolutions of various typewriter-like output devices.

Numerical Parameter Input

Both **nroff** and **troff** accept numerical input with the appended scale indicators shown in the following table, where S is the current type size in points, V is the current vertical line spacing in basic units, and C is a nominal character width in basic units.

Scale Indicator	Meaning	Number of basic units TROFF	NROFF
i	Inch	432	240
c	Centimeter	432 × 50/127	240 × 50/127
P	Pica = 1/6 inch	72	240/6
m	EM = *S* points	6 × *S*	*C*
n	EN = Em/2	6 × *S*	*C, same as Em*
p	Point = 1/72 inch	6	240/72
u	Basic unit	1	1
v	Vertical line space	*V*	*V*
none	Default, see below		

Actual character widths in **nroff** need not be all the same and constructed characters such as -> (→) are often extra wide. The default scaling is *ems* for the horizontally-oriented requests and functions **ll**, **in**, **ti**, **ta**, **lt**, **po**, **mc**, **h**, and **l**; Vs for the vertically-oriented requests and functions **pl**, **wh**, **ch**, **dt**, **sp**, **sv**, **ne**, **rt**, **v**, **x**, and **L**; **p** for the **vs** request; and **u** for the requests **nr**, **if**, and **ie**. All other requests ignore any scale indicators. When a number register containing an already appropriately scaled number is interpolated to provide numerical input, the unit scale indicator **u** may need to be appended to prevent an additional inappropriate default scaling.

The number *N*, may be specified in decimal-fraction form, but the parameter finally stored is rounded to an integer number of basic units.

The *absolute position* indicator, a vertical bar (|) may be prepended to a number N to generate the distance to the vertical or horizontal place N. For vertically-oriented requests and functions, *|N* becomes the distance in basic units from the current vertical place on the page or in a diversion to the the vertical place N. For all other requests and functions, *|N* becomes the distance from the current horizontal place on the input line to the horizontal place N. For example,

```
.sp |3.2c
```

will space in the required direction to 3.2 centimeters from the top of the page.

Numerical Expressions

Wherever numerical input is expected an expression involving parentheses, the arithmetic operators +, -, /, *, % (mod), and the logical operators <, >, <=, >=, = (or ==), & (and), : (or) maybe used. Except where controlled by parentheses, evaluation of expressions is left-to-right; **there is no operator precedence**. In the case of certain requests, an initial + or - is stripped and interpreted as an increment or decrement indicator respectively. In the presence of default scaling, the desired scale indicator must be attached to every number in an expression for which the desired and default scaling differ. For example, if the number register x contains 2 and the current point size is 10, then

```
.ll (4.25i+\nxP+3)/2u
```

will set the line length to 1/2 the sum of 4.25 inches + 2 picas + 30 points.

Notation

Numerical parameters are indicated in this manual in two ways. ±N means that the argument may take the forms N, +N, or -N and that the corresponding effect is to set the affected parameter to N, to increment it by N, or to decrement it by N respectively. Plain N means that an initial algebraic sign is not an increment indicator, but merely the sign of N. Generally, unreasonable numerical input is either ignored or truncated to a reasonable value. For example, most requests expect to set parameters to non-negative values; exceptions are **sp**, **wh**, **ch**, **nr**, and **if**. The requests **ps**, **ft**, **po**, **vs**, **ls**, **ll**, **in**, and **lt** restore the previous parameter value in the absence of an argument.

Single character arguments are indicated by single lower case letters and one/two character arguments are indicated by a pair of lower case letters. Character string arguments are indicated by multi-character mnemonics.

Page Control

Top and bottom margins are not automatically provided; it is conventional to define two macros and to set traps for them at vertical positions 0 (top) and -N (N from the bottom). A pseudo-page transition onto the first page occurs either when the first break occurs or when the first non-diverted text processing occurs. Arrangements for a trap to occur at the top of the first page must be completed before this transition. In the following, references to the current diversion mean that the mechanism being described works during both ordinary and diverted output (the former considered as the top diversion level).

The usable page width on a phototypesetter is about 7.54 inches, beginning about 1/27 inch from the left edge of the 8 inch wide, continuous roll paper. The physical limitations on output are output-device dependent.

Request Form	Initial Value	If No Argument	Notes#	Explanation
.pl ±N	11in	11in	9	Page length set to ±N. The internal limitation is about 75 inches in and about 136 inches in. The current page length is available in the .p register.
.bp ±N	N=1	-	3,4,9	Begin page. The current page is ejected and a new page is begun. If ±N is given, the new page number will be ±N. Also see request ns.
.pn ±N	N=1	ignored	-	Page number. The next page (when it occurs) will have the page number ±N. A pn must occur before the initial pseudo-page transition to affect the page number of the first page. The current page number is in the % register.
.po ±N	0; 26/27in	previous	2,9	Page offset. Values separated by ";" are forand respectively. The current left margin is set to ±N. The initial value provides about 1 inch of paper margin including the physical typesetter margin of 1/27 inch. In the maximum (line-length) + (page-offset) is about 7.54 inches. The current page offset is available in the .o register.
.ne N	-	N=1V	5,9	Need N vertical space. If the distance, D, to the next trap position is less than N, a forward vertical space of size D occurs, which will spring the trap. If there are no remaining traps on the page, D is the distance to the bottom of the page. If D<V, another line could still be output and spring the trap. In a diversion, D is the distance to the diversion trap, if any, or is very large.
.mk R	none	internal	5	Mark the current vertical place in an internal register (both associated with the current diversion level), or in register R, if given. See rt request.
.rt ±_N	none	internal	5,9	Return upward only to a marked vertical place in thecurrentdiversion. If _N (w.r.t. currentplace) is given, the place is _N from the top of the page or diversion or, if N is absent, to a place marked by a previous mk. Note that the sp request (5^3) may be used in all cases instead of rt by spacing to the absoluteplace stored ina explicit register; e. g. using the sequence .mkR ... .sp nRu.

Text Filling and Adjusting

Normally, words are collected from input text lines and assembled into an output text line until some word doesn't fit. An attempt is then made to hyphenate the word in effort to assemble a part of it into the output line. The spaces between the words on the output line are then increased to spread out the line to the current line length minus any current indent. A word is any string of characters delimited by the space character or the beginning/end of the input line. Any adjacent pair of words that must be kept together (neither split across output lines nor spread apart in the adjustment process) can be tied together by separating them with the unpaddable space character backslash space (\ ). The adjusted word spacings are uniform in **troff** and the minimum interword spacing can be controlled with the ss request. In **nroff**, they are normally nonuniform because of quantization to character-size spaces; however, the command line option -e causes uniform spacing with full output device resolution. Filling, adjustment, and hyphenation can all be prevented or controlled. The text length on the last line output is available in the .n register, and text base-line position on the page for this line is in the nl register. The textbase-line high-water mark (lowest place) on the current page is in the .h register.

An input text line ending with ., ?, or ! is taken to be the end of a sentence, and an additional space character is automatically provided during filling. Multiple inter-word space characters found in the input are retained, except for trailing spaces; initial spaces also cause a break.

When filling is in effect, a \p may be imbedded or attached to a word to cause a break at the end of the word and have the resulting output line spread out to fill the current line length.

A text input line that happens to begin with a control character can be made to not look like a control line by prefacing it with the non-printing, zero-width filler character \&. Still another way is to specify output translation of some convenient characterinto the control character using tr.

Interrupted Text

The copying of an input line in nofill (non-fill) mode can be interrupted by terminating the partial line with a \c. The next encountered input text line will be considered to be a continuation of the same line of input text. Similarly, a word within filled text may be interrupted by terminating the word (and line) with \c; the next encountered text will be taken as a continuation of the interrupted word. If the intervening control lines cause a break, any partial line will be forced out along with any partial word.

Request Form	Initial Value	If No Argument	Notes#	Explanation
.br	-	-	4	Break. The filling of the line currently being collected is stopped and the line is output without adjustment. Text lines beginning with space characters and empty text lines (blank lines) also cause a break.
.fi	fill on	-	4,6	Fill subsequent output lines. The register .u is 1 in fill mode and 0 in nofill mode.
.nf	fill on	-	4,6	Nofill. Subsequent output lines are neither filled nor adjusted. Input text lines are copied directly to output lines without regard for the current line length.
.ad c	adj,both	adjust	6	Line adjustment is begun. If fill mode is not on, adjustment will be deferred until fill mode is back on. If the type indicator c is present, the adjustment type is changed as shown in the following table.

Indicator	Adjust Type
l	adjust left margin only
r	adjust right margin only
c	center
b or **n**	adjust both margins
absent	unchanged

.na	adjust	-	6	No adjust. Adjustment is turned off; the right margin will be ragged. The adjustment type for ad is not changed. Output line filling still occurs if fill mode is on.
.ce N	off	N=1	4,6	Center the next N input text lines within the current (line-length minus indent). If N=0, any residual count is cleared. A break occurs after each of the N input lines. If the input line is too long, it will be left adjusted.

Vertical Spacing

Base-line Spacing

The vertical spacing (V) between the base-lines of successive output lines can be set using the vs request with a resolution of 1/144 inch = 1/2 point in **troff**, and to the output device resolution in **nroff**. V must be large enough to accommodate the character sizes on the affected output lines. For the common type sizes (9-12 points), usual typesetting practice is to set V to 2 points greater than the point size; default is 10-point type on a 12-point spacing. The current V is available in the .v register. Multiple-V line separation (e.g. doublespacing) may be requested with ls.

Extra Line-space

If a word contains a vertically tall construct requiring the output line containing it to have extra vertical space before and/or after it, the extra-line-space function \x'N' can be imbedded in or attached to that word. In this and other functions having a pair of delimiters around their parameter (here'), the delimiter choice is arbitrary, except that it can't look like the continuation of a number expression for N. If N is negative, the output line containing the word will be preceded by N extra vertical space; if N is positive, the output line containing the word will be followed by N extra vertical space. If successive requests for extra space apply to the same line, the maximum values are used. The most recently utilized post-line extra line-space is available in the .a register.

Blocks of Vertical Space

A block of vertical space is ordinarily requested using sp, which honors the no-space mode and which does not space past a trap. A contiguous block of vertical space maybe reserved using sv.

Request Form	Initial Value	If No Argument	Notes#	Explanation
.vsN	1/6in;12pts	previous	6,9	Set vertical base-line spacing size V. Transient extra vertical space available with \x'N'.
.lsN	N=1	previous	6	Line spacing set to ±N. N-1 Vs (blank lines) are appended to each output text line. Appended blank lines are omitted, if the text or previous appended blank line reached a trap position.
.sp N	-	N=1V	4,9	Space vertically in either direction. If N is negative, the motion is backward (upward) and is limited to the distance to the top of the page. Forward (downward) motion is truncated to the distance to the nearest trap. If the no-space mode is on, no spacing occurs (see ns, and rs below).
.sv N	-	N=1V	9	Save a contiguous vertical block of size N. If the distance to the next trap is greater than N, N vertical space is output. No-space mode has no effect. If this distance is less than N, no vertical space is immediately output, but N is remembered for later output. Subsequent sv requests will overwrite any still remembered N.
.os	-	-	-	Output saved vertical space. No-space mode has no effect. Used to finally output a block of vertical space requested by an earlier sv request.
.ns	space	-	4	No-space mode turned on. When on, the no-space mode inhibits sp requests and bp requests without a next page number. The no-space mode is turned off when a line of output occurs, or with rs.
.rs	space	-	4	Restore spacing. The no-space mode is turned off.
Blank text line		-	4	Causes a break and output of a blank line exactly like sp1.

Line Length and Indenting

The maximum line length for fill mode may be set with ll. The indent maybe set with in; an indent applicable to only the next output line may be set with ti. The line length includes indent space but not page offset space. The line-length minus the indent is the basis for centering with ce. The effect of ll, in, or ti is delayed, if a partially collected line exists, until after that line is output. In fill mode the length of text on an output line is less than or equal to the line length minus the indent. The current linelength and indent are available in registers .l and .i respectively. The length of three-part titles produced by tl is independently set by lt.

Request Form	Initial Value	If No Argument	Notes#	Explanation
.ll ±N	6.5 in	previous	6,9	Line length is set to ±N.
.in ±N	N=0	previous	4,6,9	Indent is set to ±N. The indent is prepended to each output line.
.ti ±N	-	ignored	4,6,9	Temporary indent. The next output text line will be indented a distance ±N with respect to the current indent. The resulting total indent may not be negative. The current indent is not changed.

Macros,Strings, Diversion, and Position Traps

Macros and Strings

A macro is a named set of arbitrary lines that may be invoked by name or with a trap. A string is a named string of characters, not including a new-line character, that may be interpolated by name at any point. Request, macro and string names share the same name list. Macro and string names may be one or two characters long and may usurp previously defined request, macro, or string names. Any of these entities may be renamed with rn or removed with rm. Macros are created by de and di, and appended to by am and da; di and da cause normal output to be stored in a macro. Strings are created by ds and appended to by as. A macro is invoked in the same way as a request; a control line beginning .xx will interpolate the contents of macro xx. The remainder of the line may contain up to nine arguments. The string sx and xx are interpolated at any desired point with *x and *(xx respectively. String references and macro invocations may be nested.

Copy Mode Input Interpretation

During the definition and extension of strings and macros (not by diversion) the input is read in copy mode. The input is copied without interpretation except that:

- The contents of number registers indicated by \n are interpolated.
- Strings indicated by * are interpolated.
- Arguments indicated by \$ are interpolated.
- Concealed new-lines indicated by \(new-line) are eliminated.
- Comments indicated by \" are eliminated.
- \t and \a are interpreted as ASCII horizontal tab and SOH respectively.
- \\ is interpreted as \.
- \. is interpreted as ".".

These interpretations can be suppressed by prepending a backslash (\). For example, since \\ maps into a \, \\n will copy as \n which will be interpreted as a number register indicator when the macro or string is reread.

Arguments

When a macro is invoked by name, the remainder of the line is taken to contain up to nine arguments. The argument separator is the space character, and arguments may be surrounded by double-quotes to permit imbedded space characters. Pairs of double-quotes may be imbedded in double-quoted arguments to represent a single double-quote. If the desired arguments won't fit on a line, a concealed new-line may be used to continue on the next line.

When a macro is invoked the input level is pushed down and any arguments available at the previous level become unavailable until the macro is completely read and the previous level is restored. A macro's own arguments can be interpolated at any point within the macro with \$N, which interpolates the Nth argument ($1 \leq N \leq 9$). If an invoked argument doesn't exist, a null string results. For example, the macro xx maybe defined by

```
xx \"begin definition
Today is \\$1the \\$2.
.. \"end definition
```

and called by

```
.xx Monday 14th
```

to produce the text

```
Today is Monday the 14th.
```

Note that the \$ was concealed in the definition with a prepended backslash (\). The number of currently available arguments is in the .$ register.

No arguments are available at the top (non-macro) level in this implementation. Because string referencing is implemented as a input-level pushdown, no arguments are available from within a string. No arguments are available within a trap-invoked macro.

Arguments are copied in copy mode onto a stack where they are available for reference. The mechanism does not allow an argument to contain a direct reference to a long string (interpolated at copy time) and it is advisable to conceal string references (with an extra backslash (\)) to delay interpolation until argument reference time.

Diversions

Processed output may be diverted into a macro for purposes such as footnote processing or determining the horizontal and vertical size of some text for conditional changing of pages or columns. A single diversion trap may be set at a specified vertical position. The number registers dn and dl respectively contain the vertical and horizontal size of the most recently ended diversion. Processed text that is diverted into a macro retains the vertical size of each of its lines when reread in nofill mode regardless of the current V. Constant-spaced (cs) or emboldened (bd) text that is diverted can be reread correctly only if these modes are again or still in effect at reread time. One way to do this is to imbed in the diversion the appropriate cs or bd requests.

Diversions may be nested and certain parameters and registers are associated with the current diversion level (the top non-diversion level may be thought of as the 0th diversion level). These are the diversion trap and associated macro, no-space mode, the internally-saved marked place (see mk and rt), the current vertical place (.d register), the current high-water text base-line (.h register), and the current diversion name (.z register).

Traps

Three types of trap mechanisms are available:

- page traps
- diversion traps
- and an input-line-count traps

Macro-invocation traps may be planted using what any page position including the top. This trap position may be changed using *ch*. Trap positions at or below the bottom of the page have no effect unless or until moved to within the page or rendered effective by an increase in page length. Two traps may be planted at the same position only by first planting them at different positions and then moving one of the traps; the first planted trap will conceal the second unless and until the first one is moved. If the first one is moved back, it again conceals the second trap. The macro associated with a page trap is automatically invoked when a line of text is output whose vertical size reaches or sweeps past the trap position. Reaching the bottom of a page springs the top-of-page trap, if any, provided there is a next page. The distance to the next trap position is available in the .t register; if there are no traps between the current position and the bottom of the page, the distance returned is the distance to the page bottom.

A macro-invocation trap effective in the current diversion may be planted using dt. The .t register works in a diversion; if there is no subsequent trap a large distance is returned. For a description of input-line-count traps, see it below.

Request Form	Initial Value	If No Argument	Notes#	Explanation
.de xx yy	-	.yy=..	-	Define or redefine the macro xx. The contents of the macro begin on the next input line. Input lines are copied in copy mode until the definition is terminated by a line beginning with .yy, whereupon the macro yy is called. In the absence of yy, the definition is terminated by a line beginning with "..". A macro may contain de requests provided the terminating macros differ or the contained definition terminator is concealed; ".." can be concealed as "\\.." which will copy as "\.." and be reread as "..".
.am xx yy	-	.yy=..	-	Append to macro (append version of de).
.ds xx string	-	ignored	-	Define a string xx containing string. Any initial double-quote in string is stripped off to permit initial blanks.
.as xx string	-	ignored	-	Append string to string xx (append version of ds).

.rm xx	-	ignored	-	Remove request, macro, or string. The name xx is removed from the name list and any related storage space is freed. Subsequent references will have no effect.
.rn xx yy	-	ignored	-	Rename request, macro, or string xx to yy. If yy exists, it is first removed.
.di xx	-	end	5	Divert output to macro xx. Normal text processing occurs during diversion except that page offsetting is not done. The diversion ends when the request di or da is encountered without an argument; extraneous requests of this type should not appear when nested diversions are being used.
.da xx	-	end	5	Divert, appending to xx (append version of di).
.wh N xx	-	-	9	Install a trap to invoke xx at page position N; a negative N will be interpreted with respect to the page bottom. Any macro previously planted at N is replaced by xx. A zero N refers to the top of a page. In the absence of xx, the first found trap at N, if any, is removed.
.ch xx N	-	-	9	Change the trap position for macro xx to be N. In the absence of N, the trap, if any, is removed.
.dt N xx	-	off	5,9	Install a diversion trap at position N in the current diversion to invoke macro xx. Another dt will redefine the diversion trap. If no arguments are given, the diversion trap is removed.
.it N xx	-	off	6	Set an input-line-count trap to invoke the macro xx after N lines of text input have been read (control or request lines don't count). The text may be in-line text or text interpolated by in-line or trap-invoked macros.
.em xx	none	none	-	The macro xx will be invoked when all input has ended. The effect is the same as if the contents of xx had been at the end of the last file processed.

Number Registers

A variety of parameters are available to the user as predefined, named number registers. In addition, the user may define his own named registers. Register names are one or two characters long and do not conflict with request, macro, or string names. Except for certain predefined read-only registers, a number register can be read, written, automatically incremented or decremented, and interpolated into the input in a variety of formats. One common use of user-defined registers is to automatically number sections, paragraphs, lines, etc. A number register may be used any time numerical input is expected or desired and may be used in numerical expressions.

Number registers are created and modified using nr, which specifies the name, numerical value, and the auto-increment size. Registers are also modified, if accessed with an auto-incrementing sequence. If the registers x and xx both contain N and have the auto-increment size M, the following access sequences have the effect shown

Sequence	Effect on Register	Value Interpolated
\n*x*	none	*N*
\n(*xx*	none	*N*
\n+*x*	*x* incremented by *M*	*N*+*M*
\-*x*	*x* incremented by *M*	*N*−*M*
\n+(*xx*	*xx* incremented by *M*	*N*−*M*
\n−(*xx*	*xx* decremented by *M*	*N*−*M*

When interpolated, a number register is converted to decimal (default), decimal with leading zeros, lower-case Roman, upper-case Roman, lower-case sequential alphabetic, or upper-case sequential alphabetic according to the format specified by **af**.

Request Form	Initial Value	If No Argument	Notes#	Explanation
.nr R ±N	M	-	9	The number register R is assigned the value ±N with respect to the previous value, if any. The increment for auto-incrementing is set to M.
.af R c	arabic	-	-	Assign format c to register R. The available formats are:

Format	Numbering Sequence
1	0,1,2,3,4,5,...
001	000,001,002,003,004,005,...
i	0,i,ii,iii,iv,v,...
I	0,I,II,III,IV,V,...
a	0,a,b,c,...,z,aa,ab,...zz,aaa,...
A	0,A,B,C...,Z,AA,AB,..,ZZ,AAA,...

An arabic format having N digits specifies a field width of N digits. The read-only registers and the width function are always arabic.

.rr R	-	ignored	-	Remove register R. If many registers are being created dynamically, it may become necessary to remove no longer used registers to recapture internal storage space for newer registers.

Tabs, Leaders, and Fields

Tabs and Leaders

The ASCII horizontal tab character and the ASCII SOH (hereafter known as the leader character) can both be used to generate either horizontal motion or a string of repeated characters. The length of the generated entity is governed by internal tab stops specifiable with ta. The default difference is that tabs generate motion and leaders generate a string of periods; tc and lc offer the choice of repeated character or motion. There are three types of internal tab stops

- left adjusting
- right adjusting
- and centering

In the following table: D is the distance from the current position on the input line (where a tab or leader was found) to the next tab stop; next-string consists of the input characters following the tab (or leader) up to the next tab (or leader) or end of line; and W is the width of next-string.

Tab type	Length of motion or repeated characters	Location of *next-string*
Left	*D*	Following *D*
Right	*D*– *W*	Right adjusted within *D*
Centered	*D*– *W/2*	Centered on right end of *D*

The length of generated motion is allowed to be negative, but that of a repeated character string cannot be. Repeated character strings contain an integer number of characters, and any residual distance is prepended as motion.

Tabs or leaders found after the last tab stop are ignored, but may be used as next-string terminators. Tabs and leaders are not interpreted in copy mode. \t and \a always generate a non-interpreted tab and leader respectively, and are equivalent to actual tabs and leaders in copy mode.

Fields

A field is contained between a pair of field delimiter characters, and consists of sub-strings separated by padding indicator characters. The field length is the distance on the input line from the position where the field begins to the next tab stop. The difference between the total length of all the sub-strings and the field length is incorporated as horizontal padding space that is divided among the indicated padding places. The incorporated padding is allowed to be negative. For example, if the field delimiter is the sharp sign (#), and the padding indicator is a carot (^), *# ^xxx ^right#* specifies a right-adjusted string with the string *xxx* centered in the remaining space.

Request Form	Initial Value	If No Argument	Notes#	Explanation
.ta Nt ...	8n; 0.5in	none	6,9	Set tab stops and types. t=R, right adjusting; t=C,centering; t absent, left adjusting. Tab stops are preset in **nroff** every 8 nominal character widths. The stop values are separated by spaces, and a value preceded by + is treated as an increment to the previous stop value.
.tc c	none	none	6	The tab repetition character becomes c, or is removed specifying motion.
.lc c	.	none	6	The leader repetition character becomes c, or is removed specifying motion.
.fc a b	off	off	-	The field delimiter is set to a; the padding indicator is set to the space character or to b, if given. In the absence of arguments the field mechanism is turned off.

I/O Conventions and Character Translations

Input Character Translations

The ASCII control characters horizontal tab, SOH, and backspace are discussed elsewhere. The new-line delimits input lines. In addition, STX, ETX, ENQ, ACK, and BEL are accepted, and may be used as delimiters or translated into a graphic with tr. All others are ignored.

The escape character, backslash (\) introduces:

- escape sequences
- causes the following character to mean another character
- or to indicate some function

The backslash (\) should not be confused with the ASCII control character ESC of the same name. The escape character can be input with the sequence \\. The escape character can be changed with ec, and all that has been said about the default \ becomes true for the new escape character. \e can beused to print whatever the current escape character is. If necessary or convenient, the escape mechanism may be turned off with eo, and restored with ec.

Request Form	Initial Value	If No Argument	Notes#	Explanation
.ec c	\	\	-	Set escape character to \, or to c,if given.
.eo	on	-	-	Turn escape mechanism off.

Backspacing, Underlining, Overstriking, Etc

Unless in copy mode, the ASCII backspace character is replaced by a backward horizontal motion having the width of the space character.

nroff automatically underlines characters in the underline font, specifiable with uf, normally that on font position 2 (normally Times Italic). In addition to ft and \fF, the underline font may be selected by ul and cu. Underlining is restricted to an output-device-dependent subset of reasonable characters.

Request Form	Initial Value	If No Argument	Notes#	Explanation
.ul N	off	N=1	E	Underline in **nroff** the next N input text lines. Actually, switch to underline font, saving the current font for later restoration; other font changes within the span of a ul will take effect, but the restoration will undo the last change. Output generated by tl is affected by the font change, but does not decrement N. If N>1, there is the risk that a trap interpolated macro may provide text lines within the span; environment switching can prevent this.
.cu N	off	N=1	6	A variant of ul that causes every character to be underlined in **nroff**.
.uf F	Italic	Italic	-	Underline font set to F. In **nroff**, F may not be on position 1 (initially Times Roman).

Control Characters

Both the control character period (.) and the no- break control character single-quote (') may be changed, if desired. Such a change must be compatible with the design of any macros used in the span of the change, and particularly of any trap-invoked macros.

Request Form	Initial Value	If No Argument	Notes#	Explanation
.cc c	.	.	6	The basic control character is set to c, or reset to ".".
.c2 c	'	'	6	The nobreak control character is set to c, or reset to "'".

Output Translation

One character can be made a stand-in for another character using tr. All text processing (e.g., character comparisons) takes place with the input (stand-in) character which appears to have the width of the final character. The graphic translation occurs at the moment of output (including diversion).

Request Form	Initial Value	If No Argument	Notes#	Explanation
.tr abcd....	none	-	7	Translate a into b, c into d, etc. If an odd number of characters is given, the last one will be mapped into the space character. To be consistent, a particular translation must stay in effect from input to output time.

Transparent Throughput

An input line beginning with a \! is read in copy mode and transparently output (without the initial \!); the text processor is otherwise unaware of the line's presence. This mechanism may be used to pass control information to a post-processoror to imbed control lines in a macro created by a diversion.

Comments and Concealed New-lines

An uncomfortably long input line that must stay one line (e.g., a string definition, or nofilled text) can be split into many physical lines by ending all but the last one with the escape backslash (\). The sequence\(new-line) is always ignored-except in a comment. Comments may be imbedded at the end of any line by prefacing them with \". The new-line at the end of a comment cannot be concealed. A line beginning with \" will appear as a blank line and behave like **.sp 1**; a comment can be on a line by itself by beginning the line with .\".

Width Function

The width function \w'string' generates the numerical width of string (in basic units). Size and font changes may be safely imbedded in string, and will not affect the current environment. For example, **.ti -\w'1. 'u** could be used to temporarily indent leftward a distance equal to the size of the string "1. ".

The width function also sets three number registers. The registers st and sb are set respectively to the highest and lowest extent of string relative to the baseline; then, for example, the total height of the string is **\n(stu-\n(sbu**. In the number register ct is set to a value between 0 and 3: 0 means that all of the characters in string were short lower case characters without descenders (like e); 1 means that at least one character has a descender (like y); 2 means that at least one character is tall (like H); and 3 means that both tall characters and characters with descenders are present.

Mark Horizontal Place

The escape sequence **\kx** will cause the current horizontal position in the input line to be stored in register x. As an example, the construction **\kxword\h'nxu+2u'word** will embolden word by backing up to almost its beginning and overprinting it, resulting in **word**.

More Functions

Overstriking

Automatically centered overstriking of up to nine characters is provided by the overstrike function \o***'string'***. The characters in string are overprinted with centers aligned; the total width is that of the widest character.

Zero-width Characters

The function \z*c* will output *c* without spacing over it, and can be used to produce left-aligned overstruck combinations.

Line Drawing

The function \l'N*c*' will draw a string of repeated c's towards the right for a distance N. (\l is \(lower case L). If c looks like a continuation of an expression for N, it may insulated from N with a \&. If c is not specified, the _ (baseline rule)is used (underline character in **nroff**). If N is negative, a backward horizontal motion of size N is made before drawing the string. Any space resulting from N/(size of c) having a remainder is put at the beginning (left end) of the string. In the case of characters that are designed to be connected such as baseline-rule _, underrule _,and root-en , the remainder space is covered by over-lapping. If N is less than the width of c, a single c is centered on a distance N. As an example, a macro to underscore a string can be written

```
.de us
\\$1\l'0\(ul' ..
```

or one to draw a box around a string

```
.de bx \(br\\$1\(br\l'0\(rn'\l'0\(ul' ..
```

such that .us "underlined words" and .bx "words in a box"

The function \L'N*c*' will draw a vertical line consisting of the (optional) character **c** stacked vertically 1 line apart, with the first two characters overlapped, if necessary, to form a continuous line. The default character is the *box rule* (\(br); the other suitable character is the *bold vertical* (\(bv). The line is begun without an initial motion relative to the current base line. A positive N specifies a line drawn downward and a negative N specifies a line drawn upward. After the line is drawn no compenstaing motions are made; the instataneous baseline is at the end of the line.

The horizontal and vertical line drawing functions may be used in combination to produce large boxes. The zero-width *box-rule* and the ½-em wide *underrule* were *designed* to form corners when using 1-em vertical spacings. For example the macro

```
.de eb
.sp -1      \"compensate for next automatic base-line spacing
.nf         \"avoid possible overflowing word buffer
\h'-.5n'\L'|\\nau-1'\l'\\n(.lu+1n\(ul'\L'-|\\nau+1'\l'|0u-.5n\(ul'   \"draw box
.fi
..
```

will draw a box around some text whose beginning vertical place was saved in number register *a* (e.g. using **.mk a**) as done for this paragraph.

Hyphenation

The automatic hyphenation may be switched off and on. When switched on with **hy**, several variants may be set. A hyphenation indicator character may be imbedded in a word to specify desired hyphenation points, or may be prepended to suppress hyphenation. In addition, the user may specify a small exception word list.

Only words that consist of a central alphabetic string surrounded by (usually null) non-alphabetic strings are considered candidates for automatic hyphenation. Words that were input containing hyphens (minus) or hyphenation indicator characters - such as mother-in-law - are always subject to splitting after those characters, whether or not automatic hyphnenation is on or off.

Request Form	Initial Value	If No Argument	Notes#	Explanation
.nh	hyphenate	-	6	Automatic hyphenation is turned off.
.hyN	on,N = 1	on,N = 1	6	Automatic hyphenation is turned on for N≥1, or off for N = 0. If N = 2, last lines (ones that will cause a trap) are not hyphenated. For N = 4 and 8, the last and first two characters respectively of a word are not split off. These value are additive, i.e. N = 14 will invoke all three restrictions.
.hc c	\%	\%	6	Hyphenation indicator character is set to c or to the defalult \%. The indicator does not appear in the output.
.he word1...		ignored	-	Specify hyphenation points in words with imbedded minus signs. Version of a word with terminal *s* are implied; i.e. dig-it implies dig-its. This list is examined initially and after each suffix stripping. The space available is small-about 128 characters.

Output Line Numbering

Automatic sequence numbering of output lines may be requested with **nm**. When in effect, a three-digit, arabic number plus a 3 digit-space is prepended to output text lines. The text lines are thus offset by four digit-spaces, and otherwise retain their line length; a reduction in line length may be desired to keep 6 the right margin aligned with an earlier margin. Blank lines, other vertical spaces, and line generated by **tl** are not numbered. Numbering can be temporarily suspended with **nn**, 9 or with an **.nm** followed by a later **.nm +0**. In addition, a line number indent I, and the number-text separation S may be specified in digit-spaces. Further, it can be specified that only those line numbers that are multiples of some number M are to be printed (the others will appear as blank number files).

Request Form	Initial Value	If No Argument	Notes#	Explanation
.nm ±N M S I		off	6	Line number mode. If ±N is given, line numbering is turned on, and the next output line numbered is numbered ±N. Default values are M = 1, S = 1, and I = 0. Parameters corresponding to missing arguments are unaffected; a non-numeric argument is considered missing. In the absence of all arguments, numbering is turned off; the next line number is preserved for possible futher use in number register **ln**.
.nn N	-	N = 1	6	The next N text output lines are not numbered.

As an example, the paragraph protions of this section are numbered with M = 3: **.nm 1 3** was placed at the beginning; **.nm** was placed at the end of the first paragraph; and **.nm +0** was placed in front of this paragraph; and **.nm** finally placed at the end. Line lengths were also changed (by **\w'000'u)** to keep the right side aligned. Another example is **.nm +5 5 x 3** which turns on numbering with the line number of the next line to be 5 greater than the last number line, with M = 5, with spacing S untouched, and with the indent I set to 3.

Conditional Acceptance of Input

In the following, c is a one-character, built-in condition name, ! signifies not, N is a numerical expression, string1 and string2 are strings delimited by any non-blank, non-numeric character not in the strings, and anything represents what is conditionally accepted.

Request Form	Initial Value	If No Argument	Notes#	Explanation
.if c anything		-	-	If condition c true, accept anything as input; in multi-line case use \{anything\}
.if !c anything		-	-	If condition c false, accept anything
.if N anything		-	9	If expression N>0, accept anything
.if !N anything		-	9	If expression N≤0, accept anything
.if 'string1'string2' anything			-	If string1 identical to string2, accept anything
.if !'string1'string2' anything			-	If string1 not identical to string2, accept anything
.ie c anything		-	-	If portion of if-else; all above forms (like if)
.il anything		-	-	Else portion of if-else

The built-in condition names are:

Condition Name	True If
o	Current page number is odd
e	Current page number is even
t	Formatter is TROFF
n	Formatter is NROFF

If the condition c is true, or if the number N is greater than zero, or if the strings compare identically (including motions and character size and font), anything is accepted as input. If a ! precedes the condition, number, or string comparison, the sense of the acceptance is reversed.

Any spaces between the condition and the beginning of anything are skipped over. The anything can be either a single input line (text, macro, or whatever) or a number of input lines. In the multi-line case, the first line must begin with a left delimiter \{ and the last line must end with a right delimiter \}.

The request **ie** (if-else) is identical to **if** except that the acceptance state is remembered. A subsequent and matching **el** (else) request then uses the reverse sense of that state. **ie -el** pairs may be nested.

Some examples are:

```
.if e .tl 'Even Page %'''
```

which outputs a title if the page number is even; and

```
.ie \n%>1 \{\
'sp 0.5i
.tl 'Page %'''
'sp|1.2i\}
.el .sp|2.5i
```

which treats page 1 differently from other pages.

Environment Switching

A number of the parameters that control the text processing are gathered together into an environment, which can be switched by the user. The environment parameters are those associated with requests noting 6 in their Notes column; in addition, partially collected lines and words are in the environment. Everything else is global; examples are page-oriented parameters, diversion-oriented parameters, number registers, and macro and string definitions. All environments are initialized with default parameter values.

Request Form	Initial Value	If No Argument	Notes#	Explanation
.ev N	N=0	previous	-	Environment switched to environment $0 \leq N \leq 2$. Switching is done in push-down fashion so that restoring a previous environment must be done with .ev rather than specific reference.

Insertions From the Standard Input

The input can be temporarily switched to the system's standard input with **rd**, which will switch back when two newlines in a row are found (the extra blank line is not used). This mechanism is intended for insertions in form-letter-like documentation. On HP-UX, the standard input can be the user's keyboard, a pipe, or a file.

Request Form	Initial Value	If No Argument	Notes#	Explanation
.rd prompt	-	prompt = BEL	-	Read insertion from the standard input until two newlines in a row are found. If the standard input is the user's keyboard, prompt (or a BEL) is written onto the user's terminal. rd behaves line a macro, and arguments may be placed after prompt.
.ed	-	-	-	Exit from **nroff**. Text processing is terminated exactly as if all input had ended.

If insertions are to be taken from the terminal keyboard while output is being printed on the terminal, the command line option **-q** will turn off the echoing of keyboard input and prompt only with **BEL**. The regular input and insertion input cannot simultaneously come from the standard input.

As an example, multiple copies of a form letter may be prepared by entering the insertions for all the copies in one file to be used as the standard input, and causing the file containing the letter to reinvoke itself using **nx**; the process would ultimately be ended by an **ex** in the insertion file.

Input/Output File Switching

Request Form	Initial Value	If No Argument	Notes#	Explanation
.so filename		-	-	Switch source file. The top input (file reading) level is switched to filename. The effect of an so encountered in a macro is not felt until the input level returns to the file level. When the new file ends, input is again taken from the original file. so's may be nested.
.nx filename		end-of-file	-	Next file is filename. The current file is considered ended, and the input is immediately switched to filename.
.pi program		-	-	Pipe output to program (**nroff** only). This request must occur before any printing occurs. No arguments are transmitted to program.

Miscellaneous

Request Form	Initial Value	If No Argument	Notes#	Explanation
.mc c N	p	off	3,9	Specifies that a margin character c appear a distance N to the right of the right margin after each non-empty text line (except those produced by tl). If the output line is too-long (as can happen in nofil mode) the character will be appended to the line. In N is not given, the previous N is used; the initial N is 0.2 inches in **nroff**. The margin character used with this paragraph was a 12-point box-rule.
.tm string	-	newline	-	After skipping initial blanks, string (rest of the line) is read in copy mode and written on the user's terminal.
.ig yy	-	.yy=..	-	Ignore input lines. ig behaves exactly like de except that the input is discarded. The input is read in copy mode, and any auto-incremented registers will be affected.
.pm t	-	all	-	Print macros. The names and sizes of all of the defined macros and strings are printed on the user's terminal; if t is given, only the total of the sizes is printed. The sizes is given in blocks of 128 characters.
.fl	-	-	4	Flush output buffer. Used in interactive debugging to force output.

Output and Error Messages

The output from **tm**, **pm** and the prompt **rd**, as well as various error messages are written onto HP-UX's standard message output. The latter is different from the standard output, where **nroff** formatted output goes. Be default, both are written onto the user's terminal, but they can be independently redirected.

Various error conditions may occur during the operation of **nroff**. Certain less serious errors having only local impact do not cause processing to terminate. Two examples are word overflow, caused by a word that is too large to fit into the word buffer; (in fill mode), and line overflow, caused by an output line that grew too large to fit in the line buffer; in both cases; a message is printed, the offending excess is discarded, and the affected word or line is marked at the point of truncation with an asterisk (*) in **nroff**. The philosophy is to continue processing, if possible, on the grounds that output useful for debugging may be produced. If a serious error occurs, processing terminates, and an appropriate message is printed. Examples are the inability to create, read, or write files, and the exceeding of certain internal limits that make future output unlikely to be useful.

Tutorial Examples

Introduction

Although **nroff** has, by design, a syntax reminiscent of earlier text processors with the intent of easing their use, it is almost always necessary to prepare at least a small set of macro definitions to describe most documents. Such common formatting needs such as page margins and footnotes are deliberately not built into **nroff**. Instead, the macro and string definition, number register, diversion, environment switching, page-position trap, and conditional input mechanisms provide the basis for user-defined implementations.

The examples to be discussed are intended to be useful and somewhat realistic, but won't necessarily cover all relevant contingencies. Explicit numerical parameters are used in the examples to make them easier to read and to illustrate typical values.

Page Margins

Header and footer macros are usually defined to describe the top and bottom areas respectively. A trap is planted at page position 0 for the header, and at -N (N from the page bottom) for the footer. The simplest such definitions might be:

```
.de hd                  \"define header
'sp 1i
..                      \"end definition
.de fo                  \"define footer
'bp
..                      \"end definition
.wh o hd
.wh -li fo
```

which provide blank 1 inch top and bottom margins. The header will occur on the first page, only if the definition and trap exist prior to the initial pseudo-page transition. In fill mode, the output line that springs the footer trap was typically forced out because some part or whole word didn't fit on it. If anything in the footer and header that follows causes a break, that word or part word will be forced out. In this and other examples, requests like **bp** and **sp** that normally cause breaks are invoked using the no-break control character single-quote (') to avoid this. When the header/footer design contains material requiring independent text processing, the environment may be switched, avoiding most interaction with the running text.

A more realistic example would be:

```
.de hd                          \"header
.if t .tl '\(rn'\(rn'           \"troff cut mark
.if\\n%>1 \{\
'sp0.5i-1                       \"tl base at 0.5i
.tl "-%-"                       \"centered page number
.ps                             \"restore size
.ft                             \"restore font
.vs \}                          \"restore vs
'sp1.0i                         \"space to 1.0i
.ns                             \"turn on no-space mode
..
.de fo                          \"footer
.ps 10                          \"set footer/header size
.ft R                           \"set font
.vs 12p                         \"set base-line spacing
.if \\n%=1 \{\
'sp \n(.pu-0.5i-1               \"tl base 0.5i up
.tl "-%-"\}                     \"first page number
'bp
..
.wh 0 hd
.wh -1i fo
```

which sets the size, *fon*, and base-line spacing for the header/footer material, and ultimately restores them. The material in this case is a page number at the bottom of the first page and at the top of the remaining pages. The *sp*'s refer to absolute positions to avoid dependence on the base-line spacing. Another reason for this in the footer is that the footer is invoked by printing a line whose vertical spacing swept past the trap position by possibly as much as the base-line spacing. The no-space mode is turned on at the end of *hd* to render ineffective accidental occurrences of *sp* at the top of the running text.

The above method of restoring size, font, etc. presupposes that such requests (that set previous value) are not used in the running text. A better scheme is save and restore both the current and previous values as shown for size in the following:

```
.de fo
.nr sl\\n(.s            \"current size
.ps
.nr se \\n(.s           \"previous size
. . . .                 \"rest of footer
..
.de hd
. . . .                 \"header stuff
.ps\\n(s2               \"restore previous size
.ps\\n(s1               \"restore current size
..
```

Page numbers may be printed in the bottom margin by a separate macro triggered during the footer's page ejection:

```
.de bn                  \"bottom number
.tl"-%-"                \"centered page number
..
.wh -0.5i-1v bn         \"tl base 0.5i up
```

Paragraphs and Headings

The housekeeping associated with starting a new paragraph should be collected in a paragraph macro that, for example, does the desired preparagraph spacing, forces the correct font, size, base-line spacing, and indent, checks that enough space remains for more than one line, and requstss a temporary dent.nt

```
.de pg                  \"paragraph
.br                     \"break
.ft R                   \"force font
.ps 10                  \"size
.vs 12p                 \"spacing
.in 0                   \"and indent
.sp 0.4                 \"prespace
.ne 1+\\n(.Vu           \"want more than one line
.ti 0.2i                \"temp indent
```

The first break in **pg** will force out any previous partial lines, and must occur before the **vs**. The forcing of font, etc. is partly a defense against prior error and partly to permit things like section heading macros to set parameters only once. The prespacing parameter is suitable for TROFF; a larger space, at least as big as the outpt tdevice vertal alresolution, would be more suitable in **nroff**. The choice of remaining space to test for in the **ne** is the smallest amount greater than one line (the .V is the available vertical resolution).

A macro to automatically number section headings might look like:

```
.de sc                  \"section
. ---                   \"force font, etc.
.sp 0.4                 \"prespace
.ne 2.4+  \\n(.Vu       \"want 2.4+ lines
.fi
\\n+S
..
.nr kS 0 1              \"init S
```

The usage is **.sc**, followed by the section heading text, followed by **.pg**. The **ne** test value includes one line of heading, 0.4 line in the following **pg**, and one line of the paragraph text. A word consisting of the next section number and a period is produced to begin the heading line. The format of the number may be set by **af**.

Another common form is the labeled, indented paragraph, where the label protrudes left into the indent space

```
.de lp                          \"labeled paragraph
.pg
.in 0.5i                        \"paragraph indent
.ta 0.2i 0.5i                   \"label, paragraph
.ti 0
\t\\$1\t\c                      \"flow into paragraph
..0
```

The intended usage is ".lp label"; label will begin at 0.2 inch, and cannot exceed a length of 0.3inch without intruding into the paragraph. The label could be right adjusted against 0.4-inch by setting the tabs instead with**.ta 0.4iR 0.5i**. The last line of *lp* ends with \c so that it will become a part of the first line of the text that follows.

Multiple Column Output

The production of multiple column pages requires the footer macro to decide whether it was invoked by other than the last column, so that it will begin a new column rather than produce the bottom margin. The header can initialize a column register that the footer will increment and test. The following is arranged for two columns, but is easily modified for more.

```
.de hd                  \"header
. ---
.nr cl 0 1              \"init column count
.mk                     \"mark top of text
..
.de to                  \"footer
.ie\\n+(cl<2\{\
.po +3.4i               \"next column:3.1+0.3
.rt                     \"back to mark
.el \{\
.po\\nMu                \"restore left margin
. ---
.ll 3.1i                \"column width
.nr M\\n(.o             \"save left margin
```

Typically a portion of the top of the first page contains full width text; the request for the narrower line length, as well as another **.mk** would be made where the two column output was to begin.

Footnote Processing

The footnote mechanism to be described is used by imbedding the footnotes in the input text at the point of reference, demarcated by an initial **.fn** and a terminal **.ef**:

```
.fn
Footnote text and control lines....
.ef
```

In the following, footnotes are processed in a separate environment and diverted for later printing in the space immediately prior to the bottom margin. There is provision for the case where the last collected footnote doesn't completely fit in the available space.

```
.de hd \"header
. ---
.nr x 0 1                \"init footnote count
.nr y 0-\\nb             \"current footer place
.ch fo -\\nbu            \"reset footer trap
..
.de fo                   \"footer
.nr dn 0                 \"zero last diversion size
.if\\nx\{\
.ev 1 \"expand footnotes in ev1
.nf \"retain vertical size
.FN \"footnotes
.rm FN \"delete it
.it"\\n(.z"fy" .di \"end overflow diversion
.nr x 0 \"disable fx
.ev \}                   \"po environment
. ---
'bp
..
.de fx                   \"process footnote overflow
.if\\nx.di fy            \"divert overflow
..
.de fn                   \"start footnote
.da FN                   \"divert (append) footnote
.ev 1                    \"in environment 1
.if\\n+x=1 .fs           \"if first, include separator
.fi                      \"fill mode
..
.de ef                   \"end footnote
.br                      \"finish output
.nr z\\n(.v              \"save spacing
```

```
.ev                     \"pop ev
.di                     \"end diversion
.nr y -\\n(dn           \"new footer position
.if\\nx=1 .nr y-(\\n(.v-\\nz)\\"uncertainty correction
.ch fo\\nyu             \"y is negative
.if(\\n(nl+1v)>(\\n(.p+\\nv)\
.ch fo\\n(nlu+1v        \"it didn't fit
..
.de fs                  \"separator
\l'1i'                  \"1 inch rule
.br
..
.de fz                  \"get leftover footnote
.fn
.nf                     \"retain vertical size
.fy                     \"where fx put it
.ef
..
.nr b 1.0i              \"bottom margin size
.wh o hd                \"header trap
.wh 12i fo              \"footer trap, temp position
.wh -\\nbu fx           \"fx at footer position
.ch fo -\\nbu           \"conceal fs with fo
```

The header **hd** initializes a footnote count register x, and sets both the current footer trap position register y and the footer trap itself to a nominal position specified in register b. In addition, if the register *dn* indicates a leftover footnote, *fz* is invoked to reprocess it. The footnote start macro *fn* begins a diversion (append) in environment 1, and increments the count x; if the count is one, the footnote separator fs is interpolated. The separator is kept in a separate macro to permit user redefinition. The footnote end macro ef restores the previous environment and ends the diversion after saving the spacing size in register z. y is then decremented by the size of the footnote, available in dn; then on the first footnote, y is further decremented by the difference in vertical base-line spacings of the two environments, to prevent the late triggering the footer trap from causing the last line of the combined footnotes to overflow. The footer trap is then set to the lower (on the page) of y or the current page position (nl) plus one line, to allow for printing the reference line. If indicated by x, the footer fo rereads the footnotes from FN in nofill mode in environment 1, and deletes FN. If the footnotes were too large to fit,the macro fx will be trap-invoked to redivert the overflow into fy, and the register dn will later indicate to the header whether fy is empty. Both fo and fx are planted in the nominal footer trap position in an order that causes fx to be concealed unless the fo trap is moved. The footer then terminates the overflow diversion, if necessary, and zeros x to disable fx, because the uncertainty correction together with a not-too-late triggering of the footer can result in the footnote rereading finishing before reaching the fx trap.

A good exercise for the student is to combine the multiple-column and footnote mechanisms.

The Last Page

After the last input file has ended, and invoke the end macro, if any, and when it finishes, eject the remainder of the page. During the eject, any traps encountered are processed normally. At the end of this last page, processing terminates unless a partial line, word, or partial word remains. If it is desired that another page be started, the end-macro

```
.de en                    \"end-macro
\c
'bp
..
.em
en
```

will deposit a null partial word, and effect another last page.

Table of Contents

MM: Memorandum Macros

MM
Memorandum Macros

Introduction

This tutorial is a guide and reference manual for users of Memorandum Macros (MM). These macros provide a general-purpose package of text formatting macros for use with the HP-UX text formatters ***nroff*** and ***troff*** (see *HP-UX Reference* for more details). References of the form *name*(<n>) point to page *name* of section <n> of the *HP-UX Reference*.

Conventions

Each section of this tutorial explains a single facility of MM. In general, the earlier a topic is discussed, the more necessary the information is for most users. Some of the later sections can be completely ignored if MM defaults are acceptable. Likewise, each section progresses from general case to special-case facilities. It is recommended that you read a section in detail only to point where there is enough information to obtain the desired format, then skim the rest of the section because some details may be of limited use.

In the synopses of macro calls, square brackets ([ ]) surrounding an argument indicate that it is optional. Ellipses (. . .) show that the preceding argument can appear more than once.

Since there are large differences in certain aspects of *nroff* and *troff* operation, in cases where the *nroff* formatter output differs from *troff*, *nroff* is explained first, followed by *troff* in parentheses. For example,

The title is underlined (italic).

means that the title is underlined by the *nroff* formatter, and italicized by the *troff* formatter.

Document Structure

Input for a document to be formatted with the MM text-formatting macro package has four major segments, any of which can be omitted. When present, the segments **must** appear in the following order:

1. **Parameter setting segment** sets the general style and appearance of the document. You can control page width, margin justification, numbering styles for heading and lists, page headers and footers, and many other properties of the document. You can also add macros or redefine existing ones. This segment can be omitted entirely if you are satisfied with default values. It produces no actual output, but performs only the formatter setup for the rest of the document.
2. **Beginning Segment** includes those items that occur only once, at the beginning of a document; e.g., title, author's name, and date.

3. **Body segment** is the actual text of the document. It may be as small as a single paragraph or consist of hundreds of pages. It may have a hierarchy of headings up to seven levels deep (see Paragraphs and Headings section). Headings are automatically numbered (if desired), and can be saved to generate the table of contents. Five additional levels of subordination are provided by a set of list macros for automatic numbering, alphabetical sequencing, and "marking" of list items (see Lists section). The body can contain various types of displays, tables, figures, references, and footnotes (described in later sections).
4. **Ending segment** contains those items that occur only once at the end of a document such as signatures and lists of notations (such as "Copy to:" lists). Certain macros can be invoked here to print information that is wholly or partially derived from the rest of the document, such as the cover sheet or table of contents.

Existence and size of these four segments varies widely, depending on document type and individual needs. Although a specific item (such as date, title, author names, etc.) may differ depending on the document, there is a uniform way of typing it into an input text file.

Input Text Structure

In order to make it easy to edit or revise input file text at a later time,

- Keep input lines short,
- Break lines at the end of clauses,
- Begin each new sentence on a new line.

Definitions

Formatter refers to either the *nroff* or *troff* text-formatting program.

Requests are built-in commands recognized by the formatters. Although you should seldom need to use these requests directly, you will encounter occasional references to some of the requests. For example, the request

```
.SP
```

inserts a blank line in the output at the where the request appears in the input text file.

Macros are named collections of requests. Each macro is an abbreviation for a collection of requests that would otherwise require repetition. The MM package supplies many macros, and you can define still others if you need them.

Table 1 at the end of this tutorial is an alphabetical list of macro names used by MM. The first line of each item lists the name of the macro, a brief description, and a reference to the where the macro is described. The second line illustrates a typical call to the macro.

Strings provide character variables, each of which names a string of characters. Strings are often used in page headers, page footers, and lists. These registers share the pool of names used by requests and macros. A string can be given a value via the *.ds* (define string) request. The string's value can be obtained by referencing its name, preceded by "`\*`" (for 1-character string names) or "`\*(`" (for 2-character string names). For example, the string *DT* in MM normally contains the current date, thus the input line

```
Today is \*(DT.
```

could produce the following output:

```
Today is July 4, 1984
```

The current date can be replaced by a command such as

```
.ds DT 01/01/85
```

by invoking a macro designed for that purpose (see memorandum and Released-paper Style Documents section for date changes). Table 2 at the end of this tutorial lists the string names used by MM. A brief description, documentation location reference, and initial (default) values are listed for each.

Number registers fill the role of integer variables. These registers are used as flags and for arithmetic and automatic numbering. A register can be set to a specific value by using a *.nr* request and be referenced by preceding its name with `\n` (for 1-character names) or `\n(` (for 2-character names). For example, the following line sets the value of register *d* to one more than the value of register *dd*:

```
.nr d 1+\n(dd
```

Table 3 is an alphabetical list of number register names including a brief description, documentation location reference, initial (default) value, and acceptable range of values ([m:n] means values from m through n) for each.

Naming conventions are explained in the section on Extending and Modifying MM Macros near the end of this tutorial.

Usage

This section describes how to access MM, illustrates HP-UX command lines appropriate for various output devices, and describes command line flags for the MM text formatting macro package.

The *mm* Command

The *mm*(1) command can be used to prepare documents through the *nroff* formatter and MM (*nroff* is invoked with the *−cm* flag active). The *mm* command has options to specify preprocessing by *tbl* and/or by *neqn* and for postprocessing by various output filters. Any arguments or flags that are not recognized by the *mm* command (such as −rC3) are passed to the *nroff* formatter or to MM as appropriate. The following options can occur in any order, but must appear before the file names:

Option	Meaning
`-e`	Invokes *neqn*. Also causes *neqn* to read */usr/pub/eqnchar* [(see *eqnchar*(7)].
`-t`	Invokes *tbl(1)*.
`-c`	Invokes *col(1)*.
`-E`	Invokes *−e* option of tthe *nroff* formatter.
`-y`	Uses −mm (uncompacted macros) instead of −cm.
`-12`	Use 12-pitch instead. Set pitch switch on terminal to 12, if necessary.
`-T450`	Output to DASI 450 (default terminal type unless $TERM is set [see *sh*(1)]). Equivalent to `-T1620`.
`-T450-12`	Output to DASI 450 in 12-pitch mode.
`-T300`	Output to DASI 300 terminal.
`-T300-12`	Output to DASI 300 in 12-pitch mode.
`-T300s`	Output to DASI 300S terminal.
`-T300s-12`	Output to DASI 300S in 12-pitch mode.
`-T4014`	Output to Tektronix 4014 terminal.
`-T37`	Output to TELETYPE Model 37.
`-T382`	Output to a DTC-382 terminal.
`-T4000a`	Output to a Trendata 4000A.
`-TX`	Format output for an EBCDIC line printer.
`-ThP`	Output to HP 264x terminal (implies `-c`).
`-T43`	Output to TELETYPE Model 43 (implies `-c`).
`-T40/4`	Output to TELETYPE Model 40/4 (implies `-c`).
`-T745`	Output to Texas Instruments 700 Series terminal (implies `-c`).
`-T2631`	Format output for HP 2631 printer where `-T2631-e` and `-T2631-c` can be used for expanded and compressed modes, respectively (implies `-c`).
`-TlP`	Output to a device with no reverse or partial line motions or other special features (implies `-c`).

Any `-T` option given that is not listed here does not produce an error. It is equivalent to `-TlP`.

Note

While HP-UX allows use of the options listed here, some of them refer to devices that are not included in the list of HP-UX supported peripherals. While MM may interact correctly with some or all of these devices part or all of the time, correct operation is not guaranteed. When using output devices that are not included in the current list of supported peripherals for the version of HP-UX installed on your system, proceed at your own risk.

The −cm or −mm Flag

The MM package can also be invoked by including the −cm or −mm flag as an argument to the formatter. The −cm flag causes the precompacted version of the macros to be loaded. The −mm flag causes file */usr/lib/tmac/tmac.m* to be read and processed before any other files. This action defines the MM macros, sets default values for various parameters, and initializes the formatter so it is ready to process the input text files.

Typical Command Lines

The prototype command lines are as follows (parameters are explained in the next topic):

Text without Tables or Equations

```
mm [options] file . . .
or
nroff [options] -cm file . . .

mmt [options] file . . .
or
troff [options]-cm file . . .
```

Text with Tables

```
mm -t [options] file . . .
or
tbl file . . . | nroff [options] -cm

mmt -t [options] file . . .
or
tbl file . . . | troff [options] -cm
```

Text with Equations

```
mm -e [options] file . . .
or
neqn /usr/pub/eqnchar file . . . | nroff [options] -cm

mmt -e [options] file . . .
or
eqn /usr/pub/eqnchar file . . . | troff [options] -cm
```

Text with both Tables and Equations

```
mm -t -e [options] file . . .
or
tbl file . . . | neqn /usr/pub/eqnchar - | nroff [options] -cm

mmt -t -e [options] -cm
or
tbl file . . . | eqn /usr/pub/eqnchar - | troff \ [options] -cm
```

When formatting a document with the *nroff* processor, the output should normally be processed for a specific terminal or printer type because the output may require some features that are specific to a given terminal (such as paper motion control). Some HP-UX supported terminal types and appropriate command lines for them are listed here. More information can be found in the next topic and in the *HP-UX Reference*.

Here are some examples:

- For DASI 450, 10 pitch, 6 lines/inch, 0.75-inch offset, 6-inch line length (60 characters) since this is the default terminal type, no −T option is needed (unless $TERM is set to another value). Use:

```
mm file ...
or
nroff -T450 -h -cm file ...
```

- For DASI 450, 12 pitch, 6 lines/inch, 0.75-inch offset, 6-inch line length (72 characters), use:

```
mm -12 file ...
or
nroff -T450-12 -h -cm file ...
```

- For HP 264x CRT terminal family, use:

```
mm -THP file ...
or
nroff -cm file ... | col | hp
```

- For any output device incapable of reverse paper motion and lacking hardware tab stops:

```
mm -T745 file ...
or
nroff -cm file ... | col -x
```

The *tbl*(1) and *eqn*(1)/*neqn* formatters, if needed, must be invoked as shown in the command lines illustrated earlier.

If 2-column processing is used with the *nroff* formatter, either the −c option must be specified to *mm*(1) [*mm*(1) uses *col*(1) automatically for some terminal 1 types], or the *nroff* output must be postprocessed by *col*(1). In the latter case, the −T37 terminal type must be specified for *nroff*, the −h option must **not** be specified, and the output of *col*(1) must be processed by the appropriate terminal filter. *mm*(1) with the `-c` option handles all this automatically.

Parameters Set From Command Line

Number registers are commonly used within MM to hold parameter values that control various aspects of output style. Many of these values can be changed within the text files by using *.nr* requests. In addition, some of these registers can be set from the command line. This is a useful feature for those parameters that should not be permanently embedded within the input text. If used, the number registers (with the possible exception of the register *P* below) must be set on the command line (or before the MM macro definitions are processed). The number register definitions are:

Register	Definition
-rA<n>	<n> = 1 same as invoking the .AF macro without an argument
-rC<n>	Sets type of copy (such as DRAFT) to be printed at bottom of each page. <n> = 1: OFFICIAL FILE COPY. <n> = 2: DATE FILE COPY. <n> = 3: DRAFT with single spacing and default paragraph style. <n> = 4: DRAFT with double spacing and 10-space paragraph indent.
-rD1	Sets **debug mode**. This flag requests formatter to continue processing even if MM detects errors that would otherwise cause termination. It also includes some debugging information in the default page header.
-rE<n>	Controls font of Subject/Date/From fields. <n> = 0: fields are bold (default for *troff* formatter). <n> = 1: fields are Roman font (regular-text default for *nroff* formatter).
-rL<k>	Sets length of physical page to <k> lines. For *nroff* formatter, <k> is an unscaled number representing lines. For *troff* formatter, <k> must be scaled. Default value is 66 lines per page. This flag is used, for example, when directing output to certain printers.
-rN<n>	Specifies page numbering style. <n> = 0: (default) all pages get the prevailing header. <n> = 1: page header replaces footer on page 1 only. <n> = 2: page header omitted from page 1. <n> = 3: "section-page" numbering (.FD and .RP define footnote and reference numbering in sections. <n> = 5: default page header is suppressed, but user-defined header is not affected. <n> = 6: "section-page" and "section-figure" numbering is used.

<n>	Page 1 Numbering	Remaining Pages Numbering
0	header	header
1	header replaces footer	header
2	no header	header
3	"section-page" footer	same as page 1
4	no header	no header unless .PH defined
5	"section-page" footer, "section figure"	same as page 1

Contents of the prevailing header and footer do not depend on number register's <n> value. <n> controls only whether the header (<n> = 3) or footer (<n> = 5) is printed as well as the page-number style. If header and footer are null, the value of <n> is irrelevant.

−rO<k> (Register name: uppercase "O") Offsets output <k> spaces to the right.
For *nroff*, <k> is an unscaled number representing lines or character positions.
For the *troff* formatter, <k> must be scaled.
This flag is sometimes helpful for positioning output on terminals.
Default, if not set on command line, is 1.9 cm (0.75 inch).

−rP<n> Specifies that document page numbering begin with <n>. Can also be set by using *.nr* request in input text.

−rS<n> Sets point size and vertical spacing for the document. Default <n> is 10-point type with 12-point vertical spacing, six lines per inch. Applies to *troff* formatter only.

−rT<n> Provides register settings for certain output devices.
<n> = 0: (default) no registers set.
<n> = 1: line length = 80; page offset = 3.
<n> = 2: 84 lines per page; underlining inhibited.
Applies only to *nroff*.

−rU1 Controls section head underlining (applies to *nroff* only).
Causes only letters and digits to be underlined. Otherwise, all characters (including spaces) are underlined.

−rW<k> Sets page width (line length and title length) to <k>.
For *nroff* formatter, <k> unscaled, represents character positions.
For *troff*, <k> must be scaled. Can be used to change page width from default value of 6 inches (60 characters in 10-pitch or 72 characters in 12-pitch).

Omission of −cm or −mm Flag

If a large number of arguments is required on the command line, it may be convenient to set up the first (or only) input file of a document as follows:

```
<zero or more initializations of registers in preceding list>
.so /usr/lib/tmac/tmac.m
<remainder of text>
```

In this case, you must not use the *−cm* or *−mm* flag [nor the *mm*(1) or *mmt*(1) command]. The *.so* request has the equivalent effect, but the registers must be initialized before the *.so* request because their values are meaningful only if set before macro definitions are processed. When using this method, it is best to lock into the input file only those parameters that are seldom changed. For example:

```
.nr W 80
.nr O 10
.nr N 3
.so /usr/lib/tmac/tmac.m
.H 1 "INTRODUCTION"
...
```

specifies (for the *nroff* formatter) a line length (W) of 80, a page offset (O) of 10, and "section-page" (N) numbering.

Formatting Concepts

Basic Terms

Normal formatter action fills (assembles or constructs) output lines from one or more input lines. Output lines can be justified so that both left and right margins are aligned. As lines are being filled, words can also be hyphenated when necessary. Any of these capabilities can be enabled or disabled by using *.SA*, *Hy*, and the *.nf* and *.fi* formatter requests. Disabling **fill** mode also disables justification and hyphenation.

A few formatter requests and most MM macros cause a **break**. When a break occurs, filling of the current output line ceases, the line is printed, and subsequent text begins on a new line (blank lines may be inserted before text resumes for separation of headings, paragraphs, or other items).

Formatter requests can be used with MM, but should be avoided when possible because of the consequences and side-effects each request might produce. Use MM macros (avoiding formatter requests except when necessary) because:

- Overall document style is easier to control and change when necessary.
- Complicated features such as footnotes and tables of contents are easily obtained.
- As a user, you are insulated from the peculiarities and complexities of the formatter language.

Arguments and Double Quotes

For any macro call, a null argument is an argument of zero width. Null arguments often have special meaning, and the preferred form is "". Omitting an argument is not equivalent to a null argument (as in the case of the *.MT* macro, for example). Null arguments can occur anywhere in an argument list, while omitted arguments are allowed only at the end of an argument list.

Any macro argument containing ordinary (paddable) spaces must be enclosed in double quotes. A double quote is a single character that is not to be confused with a consecutive pair of apostrophes (acute accents) or grave accents. If this convention is not carefully observed, the argument is treated as several separate arguments.

Double quotes are not permitted as part of the value of a macro argument. If it is necessary to have a macro argument value, two grave accents (` `) and/or two acute accents (' ') can be used instead. This restriction is necessary because many macro arguments are processed (interpreted) a variable number of times (as when headings are printed in text, then later in the table of contents).

Unpaddable Spaces

When output lines are justified to give an even right margin, existing spaces in a line may have additional spaces appended to them. This may distort the desired text alignment. To prevent distortion, specify an **unpaddable space**, a space that cannot be expanded during justification, by:

- Typing a backslash followed by a space (\ ). This character pair directly generates an unpaddable space.
- Sacrificing some seldom-used character [such as a tilde (~)] to be translated into a space during output.

Character translation occurs after justification, so a chosen translation character can be substituted anywhere you want to place an unpaddable space. The tilde (~) is commonly used with the translation macro for this purpose. To use the tilde in this way, insert the following at the beginning of the document:

```
.tr ~
```

If you need to print a tilde in the actual output, you can temporarily "recover" by typing:

```
.tr ~~
```

before it appears in text. To resume its use as an unpaddable space, repeat the `.tr ~` after a break or after the line containing the tilde has been output.

Note

Do not use the tilde in this fashion when the tilde is used within equations.

Hyphenation

Formatters do not perform hyphenation unless requested. To disable hyphenation in a body of text, specify:

```
.nr Hy 1
```

once at the beginning of the document input file. Hyphenation within footnotes and across page boundaries is discussed in the Footnotes section of this tutorial.

Formatters hyphenate words automatically when hyphenation is enabled. You can override automatic hyphenation and specify where hyphenation is to be allowed in certain words. You can also specify that other words are not to be hyphenated. To exercise either option, use a hypenation indicator (initially the 2-character sequence "\%"). Placing the sequence immediately before a word prevents hypenation. If the sequence occurs within a word, hyphenation can occur only where the sequence has been inserted. During formatting, the sequence is removed from the input text, and is not included in the formatter output. You can specify hyphenation in a small list of words containing a total of up to about 128 characters.

To redefine the hyphenation sequence, use the command:

```
.HC <hyphenation indicator>
```

The circumflex (^) is commonly used for this purpose by inserting the command:

```
.HC ^
```

near the beginning of the document text file (before text starts).

Note

Any word containing hyphens or dashes (also known as *em* dashes) is hyphenated (if necessary) immediately after the hyphen or dash, even if the formatter hyphenation function is not enabled.

You can construct a small list of exception words with correct hyphenation points indicated by using the *.hw* request. Here are examples using the words "computer", "program", and "knowledge" (these and other words tend to be incorrectly hyphenated by typesetting formatters):

```
.hw com-put-er
.hw prog-ram
.hw know-ledge
```

Tabs

Macros *.MT*, *.TC*, and *.CS* use the formatter tabs (*.ta*) request to set tab stops. They then restore the default tab settings (every eight characters in *nroff* formatter; every half inch in *troff* formatter) when finished. Users are responsible for tab settings other than the default values.

Default tab settings are at 9, 17, 25, ..., 161 for a total of 20 tab-stop settings. When using the *ta* command, multiple tab values are separated by commas, spaces, or any other non-numeric character. Tab stops can be set to any desired value; for example:

```
.ta 9 17 25 46 72 84 106
```

Tab characters are interpreted with respect to their position in the *input line*; not their position in the *output line*. Therefore, in general, tab characters should be used *only* in lines that are processed in no-fill (*.nf*) mode (discussed near the beginning of this section).

The table formatter (*tbl*) changes tab stops, but does not restore default tab settings.

BEL Character

The non-printing character BEL is used as a delimiter in many macros to compute the width of an argument or to delimit arbitrary text such as in page headers and footers, headings, lists, and tables. Using BEL characters in input text files, and especially in arguments to macros, produces unpredictable output.

Bullets

A bullet (•) is often obtained on a typewriter terminal by using an "o" overstruck by a "+". To maintain compatibility with the *troff* formatter, MM uses the following string to produce a bullet:

```
\*(BU
```

The bullet list (*.BL*) macro uses this string to automatically generate bullets for bullet-listed items.

Dashes, Minus Signs, and Hyphens

The *troff* formatter treats a dash, a minus sign, and a hyphen as dissimilar entities, while *nroff* treats them all identically. Therefore:

- If you use *nroff*, you can use the minus sign (-) for hypens, minus signs, and dashes.
- If you usually use *troff*, follow *troff* escape conventions.
- If you use both formatters, use care in input text preparation. Unfortunately, these graphic characters cannot be represented in a way that is both convenient and compatible with both formatters.

Here is a suggested approach:

Symbol	Action
Dash	Type `\*(EM` for each text dash for both *nroff* and *troff* formatters. This string generates an em dash in the *troff* formatter and two dashes (--) in the *nroff* formatter. Dash list (*.DL*) macros automatically generate an em dash for each list item.
Hyphen	Type - and use as-is for both formatters. The *nroff* formatter will print it as-is, while the *troff* formatter prints a - (a true hyphen).
Minus	Type `\-` for a true minus sign, regardless of formatter. The *nroff* formatter ignores the "\"; *troff* prints a true minus.

Trademark String

A trademark string `\*(Tm` is included in MM. It places the letters "TM" one-half line above the text that it follows. For example:

```
The book
.I
Introducing the UNIX
.R
\h'-1'\*(Tm
.I
System
.R
is available from the library.
```

yields:

> The book *Introducing the UNIX™ System* is available from the library.

Use of Formatter Requests

Use of formatter requests should be generally avoided because MM provides the corresponding formatting functions in a much more user-oriented and surprise-free fashion than do basic formatter requests. However, some formatter requests are useful with MM, namely the following:

Request	Function
.af	Assign format
.br	Break
.ce	Center
.de	Define macro
.ds	Define string
.fi	Fill output lines
.hw	Hyphenation of exception word
.ls	Line spacing
.nf	No filling of output lines
.nr	Define and set number register
.nx	Go to next file (does not return)
.rm	Remove macro
.rr	Remove register
.rs	Restore spacing
.so	Switch to source file and return
.sp	Space
.ta	Tab stop settings
.ti	Temporary indent
.tl	Title
.tr	Translate
.!	Escape character

The *.fp*, *.lg*, and *.ss* requests are also sometimes useful for the *troff* formatter. Use of other requests without a full understanding of their implications frequently leads to disastrous results.

Paragraphs and Headings

Paragraphs

.P<type>
One or more lines of text.

The *.P* macro is used to control paragraph style.

Paragraph Indentation

An indented or non-indented paragraph is defined by the <type> argument.

<type>	Result
0	Left-justified
1	Indent

In a left-justified paragraph, the first line begins at the left margin. The first line of an indented paragraph is indented the amount specified by the *Pi* register (default value = 5). For example, to obtain a 10-space indent, include the following line near the beginning of the file before text:

```
.nr Pi 10
```

A document input file possesses a default paragraph type obtained by including .P before each paragraph that does not follow a heading. Default paragraph type is controlled by the *Pt* number register. The initial *Pt* value of zero provides left-justified paragraphs with no first-line indent.

To force a first-line indent in all paragraphs, place this command in the beginning lines of the document input file:

```
.nr Pt 1
```

To indent the first line of all paragraphs **except** those immediately following headings, lists, and displays, use this line instead:

```
.nr Pt 2
```

Both the *Pi* and the *Pt* register must have non-zero values before any paragraphs can be indented unless indentation is forced or suppressed by a .P0 or .P1 macro request.

Note

Values specifying indentation must be unscaled, and are treated as character positions; i.e., the number of ens. The *nroff* formatter treats an en as a single character width. In the *troff* formatter, an en is a space width; (in points) equal to half the current point size for characters.

To override *Pi* and *Pt* registers, use the .P0 macro request to force left justification; .P1 to indent the first line of an individual paragraph by the amount specified by *Pi*.

If the new-paragraph request, .P, occurs inside a list, the paragraph indent is added to the current list indent.

Numbered Paragraphs

Numbered paragraphs can be produced by setting register *Np* to 1. This produces paragraphs numbered within first-level headings (for example, 1.01, 1.02, 1.03, 2.01, etc.).

To produce numbered paragraphs with the text following the first line indented such that it is aligned with the beginning of text in the first line (so that the number stands out on the left side of the paragraph), use the *.nP* macro instead of *.P*. Paragraphs are numbered within second-level headings. Here is an example:

```
.H1 "FIRST HEADING"
.H2 "Second Heading"
.nP
```
One or more lines of text.

Spacing Between Paragraphs

Number register *Ps* controls the spacing between paragraphs. Default *Ps* value is **1**, yielding one blank space (one half of a vertical space).

Numbered Headings

```
.H <level> [<heading text>] [<heading suffix>]
```
Zero or more lines of text

The *.H* macro provides seven <level>s of numbered headings. Level 1 is the highest (most significant or dominant) order; level 7 the lowest (least significant) order. The <heading suffix> argument is added to the <heading text>, and can be used for footnote marks or other information that is not to be included in the table of contents.

Note

A *.P* macro is not needed immediately after a *.H* or *.HU* because the *.H* macro performs the *.P* function. However, in order to maintain uniformity throughout the text, it is a good practice to start every paragraph with a *.P*. A *.P* immediately following a heading command is ignored.

Normal Appearance

The effect of the *.H* macro varies according to argument <level>. First-level heads are preceded by two blank lines (one vertical space). All other heads are preceded by one blank line (one half of a vertical space). Other effects are as follows:

`.H 1` <heading text>	Produces a bold-font heading followed by a single blank line (one half of a vertical space). Text following the head begins on a new line and is indented according to current paragraph type. Use of full capital letters makes the heading stand out.
`.h 2` <heading text>	Produces a bold-font heading followed by a single blank line (one half of a vertical space). Text is handled the same way as first-level heads. Initial capitals are usually used in second level heads.
`.H` <n> <heading text>	Produces an underlined (italicized) heading followed by two spaces. <n> can have any value, 3 thru 7. Text following the head begins on the same line; i.e., these are run-in headings.

Appropriate numbering and spacing (both vertical and horizontal) occur, even if the <heading text> is omitted from the *.H* macro call.

If you are satisfied with normal default operation and appearance of headings, skip to the topic "Unnumbered Headings".

Altering Appearance of Numbered Headings

Heading appearance can be readily changed by setting certain registers and strings at the beginning of the document file. Thus, document style and appearance is easily altered by changing a few lines rather than editing commands throughout the document.

Prespacing and Page Ejection

First level headings (.H 1) are normally preceded by two blank lines (one vertical space); all other headings by one-half that much. If a multi-line heading would split across page boundaries, the entire heading is moved to the next page. You can force all first-level headings to the top of the next page by using:

```
.nr Ej 1
```

in the beginning part of the document input text file. To force second- or third-level heads to begin at the top of a new page (sometimes useful in long documents), change the number at the end of the command to correspond with the largest heading-level number to be moved to the next page. For example, to start all first-, second-, and third-level heads on a new page, use:

```
.nr Ej 3
```

Spacing After Headings

Three registers control text appearance following a *.H* call:

Hb **Heading break level** defines the lowest order head (highest <n>) that causes a break (in effect terminating the heading line). Headings with higher <n> values are not separated from the text that follows the heading (run-in headings). Default value is 2.

Hs **Heading space level** defines the lowest order head (highest <n>) that is always followed by a blank line (one half of a vertical space) before subsequent text. All heading levels of value <n> or smaller are automatically followed by a blank line. Default value is 2.

Hi **Post-heading indent** defines the indent style for text that follows the heading (stand-alone headings only; does not apply to run-in headings):

Hi = 0: text is left-justified.
Hi = 1 (default): indent according to paragraph type as defined by value of *Pt.*
Hi = 2: text is aligned with beginning of first word of heading so that the heading number stands out more clearly.

For example, to place a blank line after all first, second, and third-level heads; allow no run-in headings; and to force all text following headings to be left-justified (regardless of the value of *Pt*), use the following commands in the beginning of the document input text file:

```
.nr Hs 3
.nr Hb 7
.nr Hi 0
```

Centered Headings

Use *Hc* to obtain centered headings. Headings are centered when the value of the heading level (<n>) is less than or equal to the value of *Hc* **and** the heading is a stand-alone (not a run-in) heading. Default value of *Hc* is 0.

Bold, Italic, and Underlined Headings

Control by level: Any heading that is underlined by *nroff* is italicized by *troff*. The heading font string *HF* contains seven codes that specify fonts for heading levels 1 thru 7. Acceptable codes, code interpretations, and defaults for HF codes are:

	HF Code/Function			Default for Heading Level
Formatter	**1**	**2**	**3**	**1 2 3 4 5 6 7**
nroff	no underline	underline	bold	3 3 2 2 2 2 2
troff	Roman	italic	bold	3 3 2 2 2 2 2

Default for level 1 and 2 headings is bold. Remaining levels are underlined by *nroff*, italicized by *troff*. You can alter *HF* in any combination to meet your needs. Any value omitted from the right end (less significant headings) are assumed to have a value of 1. Thus, the following request produces bold for the first five levels and Roman font for the remaining two levels when processed by *troff*.

```
.ds HF 3 3 3 3 3
```

NROFF Underlining Style: The *nroff* formatter underlines in either of two styles:

- Normal style (*.ul* request) underlines only letters and digits.
- Continuous style (*.cu* request) underlines **all** characters including spaces.

By default, MM attempts to use continuous underlining on any heading that is to be underlined and short enough to fit on a single line. If the heading is to be underlined, but does not fit on a single line, normal underlining is used instead.

Use the `-rU1` flag when invoking *nroff* to force normal underlining of all headings in a document.

Heading point sizes: Use the *HP* string (*troff* only) to specify point size for each heading level as follows:

```
.ds HP [ps1] [ps2] [ps3] [ps4] [ps5] [ps6] [ps7]
```

By default, heading text (*.H* and *.HU* is printed in the same point size as the document body except that bold stand-alone headings are printed one point size smaller than the body. *HP* is similar to *HF*, and can be specified to contain up to seven values corresponding, left-to-right, to heading levels 1 thru 7. For example,

```
.ds HP 12 12 10 10 10 10 10
```

prints first and second-level heads in 12-point size; remaining heads in 10-point. Relative values can also be specified:

```
.ds HP +2 +2 -1 -1
```

If absolute sizes are specified, they are used independent of body text point size. Relative size is based on specified text body point size, even if the body point size changes. Only values on the right can be omitted. If not present or ZERO, default is assumed.

Note

Only the heading point size is affected by *HF*. If vertical spacing (user-exit macros *.HX* or *.HZ*) is not adjusted accordingly, overprinting may result.

Marking Styles: Numerals and Concatenation

Heading marks are (usually) numerical identifiers preceding the heading text. They are printed on the same line as the heading text.

```
.HM [arg1] . . . [arg7]
```

Registers *h1* thru *H7* are counters for the seven heading levels. Register values are normally printed using Arabic numerals. You can override the default to substitue other markings for use in outlines and other document styles. This macro can have up to seven arguments; each argument a string that indicates what marking type is to be used. Allowable arguments are:

Argument	Meaning
1	Arabic (default for all levels)
0001	Arabic with enough leading zeros to get the specified number of digits
A	Uppercase alphabetic
a	Lowercase alphabetic
I	Uppercase Roman
i	Lowercase Roman

Omitted arguments are interpreted as 1. Illegal arguments have no effect.

The default heading mark construction successively appends the current mark value for each preceding heading level down to the current heading with a period between each value (for example first-level head #4, second-level head #5, third-level head #2, fourth-level head #1 produces the heading mark for the fourth level head: **4.5.2.1**). To suppress the concatenation of levels, reducing the mark to a single number followed by a period, set the heading-mark **type** register *Ht* value to 1. For example, a commonly used "outline" document style is obtained with the following sequence of requests:

```
.HM I A 1 a i
.nr Ht 1
```

Unnumbered Headings

```
.HU <heading text>
```

The *.HU* macro is a special case of .H; it is handled in the same way as .H except that no heading mark is printed. In order to preserve the hierarchical structure of headings when *.H* and *.HU* calls are intermixed, each *.HU* heading is considered to exist at the level given by register *Hu*, whose initial value is 2. Thus, in the normal case, the only difference between

```
.HU <heading text>
```

and

```
.H 2 <heading text>
```

is the printing of the heading mark for the latter. Both macros have the effect of incrementing the numbering counter for level 2 and resetting the counters for levels 3 through 7 to zero. Usually, the value of *Hu* should be set to make unnumbered headings (if any) be the highest-order (lowest <n>) headings in a document.

The .HU macro can be especially helpful in setting up appendices and other sections that may not fit well into the numbering scheme for the main body of a document.

Headings and Table of Contents

The text of headings and their corresponding page numbers can be automatically collected for a table of contents. This is accomplished by doing the following:

1. Use the content level register *Cl* to specify what level headings are to be saved, and
2. Invoke the *.TC* macro at the end of the document.

Any heading whose level is less than or equal to the value of the *Cl* register is saved and later displayed in the table of contents. The default value for the Cl register is 2; i.e., the first two levels of headings are saved.

Because of how headings are saved, it is possible to exceed the formatter's storage capacity, particularly when saving many heading levels in a large document while also processing displays and footnotes. If this happens the `Out of temp file space` formatter error message (Table 4) is issued. The only remedy is to save fewer levels and/or reduce the number of words in the headings being saved.

First-Level Headings and Page Numbering Style

Unless you specify otherwise, pages are numbered sequentially at the top of the page. For large documents, it may be desirable to use page numbering of the "section-page" form where "section" is the number of the current first-level heading. This page numbering style can be achieved by specifying the `-rN3` or `-rN5` flag on the *mm* command line. As a side effect, this also sets *Ej* to 1; i.e., each first level section begins on a new page. In this style, the page number is printed at the bottom of the page to ensure that the correct section number is printed.

User Exit Macros

Note

This topic is intended primarily for users who are accustomed to writing formatter macros.

`.HX` <dlevel> <rlevel> <heading text>
`.HY` <dlevel> <rlevel> <heading text>
`.HZ` <dlevel> <rlevel> <heading text>

Use the *.HX*, *.HY*, and *.HZ* macros to obtain a final level of control over the previously described heading mechanism. These macros are not defined by MM – they are intended to be defined by the user. The *.H* macro invokes *.HX* shortly before the actual heading text is printed; it calls *.HZ* as its last action. After *.HX* is invoked, the size of the heading is calculated. This processing causes certain features that may have been included in *.HX* (such as *.ti* for temporary indent) to be lost. After the size calculation, *.HY* is invoked so that the you can respecify these features. All-default actions occur if these macros are not defined. If you have defined *.HX*, *.HY*, or *.HZ*, the definition you supplied is interpreted at the appropriate time. Therefore, these macros can influence handling of all headings because the *.HU* macro is actually a special case of the *.H* macro.

If you originally invoked the *.H* macro, the derived level (<dlevel>) and the real level (<rlevel>) are both equal to the level given in the *.H* invocation. If you originally invoked the *.HU* macro, <dlevel> is equal to the contents of register *Hu*, and <rlevel> is 0. In both cases, <heading text> is the text of the original invocation.

By the time *.H* calls *.HX*, it has already incremented the heading counter of the specified level, produced blank lines (vertical spaces) to precede the heading, and accumulated the "heading mark" (the string of digits, letters, and periods needed for a numbered heading). When *.HX* is called, all user-accessible registers and strings can be referenced, as well as the following:

string }0	If <rlevel> is nonzero, this string contains the "heading mark". Two unpaddable spaces (to separate the mark from the heading) have been appended to this string. If <rlevel> is 0, this string is null.
register ;0	This register indicates what type of spacing is to follow the heading. A value of 0 means that the heading is run-in. A value of 1 means a break (but no blank line) is to follow the heading. A value of 2 means that a break **and** a blank line (one-half a vertical space) are to follow the heading.
string }2	If "register ;0" is 0, this string contains two unpaddable spaces that will be used to separate the (run-in) heading from the following text. If "register ;0" is nonzero, this string is null.
register ;3	This register contains an adjustment factor for the *.ne* request that is issued before the heading is actually printed. On entry to *.HX*, its value is 3 if <dlevel> = 1; otherwise it is 1. The number of lines requested by *.ne* includes a number of blank lines equal to the value contained in "register ;3" (one-half vertical space per line) plus the number of lines in the heading.

You can alter the values of }0, }2, and ;3 within *.HX*. Here are some examples of actions that can be performed by defining *.HX* to include the lines shown:

To change first-level heading mark from format *n.* to *n.0*:

```
.if\\$1=1 .ds }0\\n(H1.0\<sp>\<sp>
```

(where <sp> stands for a space)

To separate a run-in heading from the text with a period and two unpaddable spaces:

```
.if\\n(;0=0 .ds }2 .\<sp>\<sp>
```

To ensure that at least 15 lines are left on the page before printing a first-level heading:

```
.if\\$1=1 .nr ;3 15-\\n(;0
```

To add three additional blank lines before each first-level heading:

```
.if\\$1=1 .sp 3
```

To indent level 3 run-in headings by five spaces:

```
.if\\$1=3 .ti 5n
```

If temporary strings or macros are used within *.HX*, choose their names with care.

When the *.HY* macro is called after the *.ne* is issued, certain features requested in *.HX* must be repeated. For example:

```
.de HY
.if\\$1=3 .ti 5n
...
```

The *.HZ* macro is called at the end of *.H* to permit user-controlled actions after the heading is produced. In a large document, sections may correspond to chapters of a book; you may want to change a page header or footer, e.g.:

```
.de HZ
.if\\$1=1 .PF"Section\\$3"
...
```

Hints for Large Documents

For convenience, a large document is often organized into one input text file per section. If the files are numbered, it is helpful to use enough digits in the names of these files to accommodate the maximum number of sections; i.e., use suffix number 01 through 20 rather than 1 through 9 and 10 through 20.

If you want to format individual sections of long documents with the correct section numbers, it is necessary to set register *H1* to one less that the number of the section just before the corresponding `.H\!t1` call. For example, at the beginning of Section 5 of a document, insert

```
.nr H1 4
```

Note

This is not good practice. It defeats the automatic (re)numbering of sections when sections are added or deleted. Such lines should be removed as soon as possible.

Lists

This section describes different styles of lists: automatically numbered and alphabetized lists, bullet lists, dash lists, lists with arbitrary marks, and lists starting with arbitrary strings; i.e., with terms or phrases to be defined.

List Macros

In order to avoid repetitive typing of arguments that define the style or appearance of items in a list, MM provides a convenient way to specify lists. All lists share the same overall structure and are composed of the following basic parts:

- A list-initialization macro (*.AL*, *.BL*, *.DL*, *.ML*, *.RL*, or *.VL*) determines the style of list: line spacing, indentation, marking with special symbols, and numbering or alphabetizing of list items.
- One or more list-item macros (*.LI*) identifies each unique item to the system. It is followed by the actual text of the corresponding list item.
- The list-end macro (*.LE*) identifies the end of the list. It terminates the list and restores the previous indentation.

Lists can be nested up to six levels. The list-initialization macro saves the previous list status (indentation, marking style, etc.); the *.LE* macro restores it.

With this approach, the format of a list is specified only once at the beginning of the list. In addition, by building onto the existing structure, you can create your own customized sets of list macros with relatively little effort.

List-Initialization Macros

List-initialization macros are implemented as calls to the more basic *.LB* macro.

They are:

.AL	Automatically Numbered or Alphabetized List
.BL	Bullet List
.DL	Dash List
.ML	Marked List
.RL	Reference List
.VL	Variable-item List

Automatically Numbered or Alphabetized List

```
.AL [<type>] [<text indent>] [1]
```

The *.AL* macro is used to begin sequentially-numbered or alphabetized lists. If there are no arguments, the list is numbered and text is indented by *Li* (initially six) spaces from the indent in force when *.AL* was called. This leaves room for a space, two digits, a period, and two spaces before the text. Values that specify indentation must be unscaled and are treated as "character postions"; i.e., number of ens.

Spacing before the list and between list items can be suppressed by setting the list space register (*Ls*). The *Ls* register is set to the innermost list level for which spacing is done. For example:

```
.nr Ls 0
```

specifies that no spacing will occur around any list items. The default value for *Ls* is six (the maximum list nesting level).

The <type> argument can be included to obtain a different type of sequencing. Its value indicates the first element in the sequence desired. If <type> is omitted or null, <type> = 1 is assumed.

Argument	Interpretation
1	Arabic (default for all levels)
A	Uppercase alphabetic
a	Lowercase alphabetic
I	Uppercase Roman
i	Lowercase Roman

If <text indent> argument is non-null, it is used as the number of spaces from the current indent to the text; i.e., it is used instead of *Li* for this list only. If <text indent> is null, the value of *Li* is used.

If the third argument [1] is given, a blank line (one-half of a vertical space) is produced before the first item, but no blank lines are output between items in the list.

Bullet List

```
.BL [<text indent>] [1]
```

The *.BL* macro begins a bullet list. Each list item is marked by a bullet (•) followed by one space.

If the <text indent> argument is non-null, it overrides the default indentation (the amount of paragraph indentation as given in the *Pi* register). In the default case, the text of bullet and dash lists lines up with the beginning of the first line of indented paragraphs.

If the second argument [1] is specified, no blank lines separate items in the list.

Dash List

```
.DL [<text indent>] [1]
```

The *.DL* macro is indentical to *.BL* except that a dash is used as the list item mark instead of a bullet.

Marked List

```
.ML <mark> [<text indent>] [1]
```

The *.ML* macro is much like *.BL* and *.DL* macros but expects you to specify an arbitrary mark which can consist of more than a single character.

Text is indented <text indent> spaces if <text indent> is not null; otherwise, the text is indented one more space than the width of <mark>.

If the third argument [1] is specified, no blank lines separate items in the list.

Note

<Mark> must not contain ordinary (paddable) spaces because alignment of items will be lost if the right margin is justified.

Reference List

```
.RL [<text indent>] [1]
```

A *.RL* macro call begins an automatically-numbered list in which the numbers are enclosed by square brackets ([]).

If the text-indent argument is non-null, it is used as the number of spaces from the current indent to the beginning of text, i.e., it is used instead of *Li* for this list only. If the text-indent argument is omitted or null, the value of *Li* used.

If the second argument [1] is specified, no blank lines separate items in the list.

Variable-Item List

```
.VL <text indent> [<mark indent>] [1]
```

When a list begins with a *.VL* macro, there is effectively no current *mark*; it is expected that each *.LI* will provide its own mark. This form is commonly used to display definitions of terms or phrases.

- <Text indent> provides the distance from current indent to beginning of the text.
- <Mark indent> produces the number of spaces from current indent to the beginnning of the mark, and defaults to 0 if omitted or null.

If the third argument [1] is specified, no blank lines separate items in the list.

Here is an example of *.VL* macro usage:

```
.tr~
.VL 20 2
.LI mark~1
Here is a description of mark 1;
"mark 1" of the .LI line contains a tilde
translated to an unpaddable space in order
to avoid extra space between
"mark" and "1".
.LI second~mark.
This is the second mark also using a tilde translated to an unpaddable
space.
.LI third~mark~longer~than~indent:
This item shows the effect of a long mark: one space separates the mark from
 the text.
.LI~
This item effectively has no mark following the .LI is translated into a
space.
.LE
```

when formatted yields:

mark 1 Here is a description of mark 1; "mark 1" of the .LI line contains a tilde translated to an unpaddable space in order to avoid extra spaces between "mark" and "1".

second mark This is the second mark also using a tilde translated to an unpaddable space.

third mark longer than indent: This item shows the effect of a long mark; one space separates the mark from the text.

This item effectively has no mark because the tilde following the .LI is translated into a space.

In the preceding example, the tilde argument on the last `.LI` is required. Otherwise, a "hanging indent" would have been produced. To produce a "hanging indent", use *.VL* and call *.LI* with no arguments or with a null first argument. For example:

```
.VL 10
.LI
Here is some text to show a hanging indent.
The first line of text is at the left margin.
The second is indented 10 spaces.
.LE
```

when formatted yields:

Here is some text to show a hanging indent. The first line of text is at the left margin. The second is indented 10 spaces.

Note

<Mark> must not contain ordinary (paddable) spaces because alignment of items will be lost if the right margin is justified.

List-Item Macro

`.LI [<mark>] [1]`
one or more lines of text that make up the list item.

The *.LI* macro is used with all lists and for each list item. It normally creates a single blank line (one-half a vertical space) before its list item although this can be suppressed.

- If no arguments are given, *.LI* labels the item with the current mark which is specified by the most recent list-initialization macro.
- If a single argument is given, that argument is output instead of the current mark.
- If two arguments are given, the first argument becomes a prefix to the current mark thus allowing the user to emphasize one or more items in a list. One unpaddable space is inserted between the prefix and the mark.

For example:

```
.BL 6
.LI
This is a simple bullet item.
.LI+
This replaces the bullet with a "plus".
.LI + 1
This uses a "plus" as prefix to the bullet.
.LE
```

when formatted yields:

- This is a simple bullet item.

\+ This replaces the bullet with a "plus".

\+• This uses "plus" as prefix to the bullet.

Note

The mark must not contain ordinary (paddable) spaces because alignment of items will be lost if the right margin is justified.

If the current mark (in the current list) is a null string **and** the <mark> argument of *.LI* is omitted or null, the resulting effect is that of a "hanging indent"; i.e., the first line of the subsequent text is moved to the left, starting at the same place where <mark> would have started.

List-End Macro

```
.LE [1]
```

The *.LE* macro restores the state of the list to that existing just before the most recent list-initialization macro call. If the optional argument is given, *.LE* outputs a blank line (one-half of a vertical space). This option should generally be used only when the *.LE* is followed by running text but not when followed by a macro that produces blank lines of its own such as *.P*, *.H*, or *.LI*.

.H and *.HU* automatically clear all list information. You can omit the *.LE* macros that would normally occur just before either heading macro to prevent receiving an `LE:mismatched` error message, but such a practice is not recommended because errors will result if the list text is separated from the heading at some later time (such as when additional text is inserted).

Example of Nested Lists

Here is an example of input for several lists. The corresponding output follows. The *.AL* and *.DL* macro calls shown are examples of list-initialization macros. Input text is:

```
.AL
.LI
This is alphabetized list item A.
This text shows the alignment of the second line
of the item.
Notice the text indentations and alignment of left
and right margins.
.AL
.LI
This is numbered item 1.
This text shows the alignment of the second line
of the item.
The quick brown fox jumped over the lazy dog's back.
.DL
.LI
This is a dash item.
This text shows the alignment of the second line
of the item.
The quick brown fox jumped over the lazy dog's back.
.LI + 1
This is a dash item with a "plus" as prefix.
This text shows the alignment of the second line of the item.
The quick brown fox jumped over the lazy dog's back.
.LE
.LI
This is numbered item 2.
.LE
.LI
This is another alphabetized list item B.
This text shows the alignment of the second line
of the item.
The quick brown fox jumped over the lazy dog's back.
.LE
.P
This paragraph follows a lst item and is aligned with
the left margin.
A paragraph following a list resumes the normal line
length and margins.
```

The output is:

A. This is alphabetized list item A. This text shows the alignment of the second line of the item. Notice the text indentions and alignment of left and right margins.
   1. This is numbered item 1. This text shows the alignment of the second line of the item. The quick brown fox jumped over the lazy dog's back.

      — This is a dash item. This text shows the alignment of the second line of the item. The quick brown fox jumped over the lazy dog's back.

      +— This is a dash item with a "plus" as prefix. This text shows the alignment of the second line of the item. The quick brown fox jumped over the lazy dog's back.

   2. This is another alphabetized list item

B. This text shows the alignment of the second line of the item. The quick brown fox jumped over the lazy dog's back.

This paragraph follows a list item and is aligned with the left margin. A paragraph following a list resumes the normal line length and margins.

List-Begin Macro and Customized Lists

`.LB` <text indent> <mark-indent> <pad> <type> [<mark>] [<LI space>]\[<LB space>]

List-initialization macros described previously suffice for almost all cases. However, you may occasionally need more control over the layout of lists, obtainable by using the basic list-begin macro (*.LB*). *.LB* is used by the other list-initialization macros, and its arguments are as follows:

- The <text indent> argument provides the number of spaces that text is to be indented from the current indent. Normally, this value is taken from the *Li* register (for automatic lists) or from the *Pi* register (for bullet and dash lists).
- The combination of <mark indent> and <pad> arguments determines the placement of the <mark>. The <mark> is placed within an area (called mark area) that starts <mark indent> spaces to the right of the current indent and ends where the text begins (i.e., ends <text indent> spaces to the right of the current indent). <Mark indent> is usually set to 0.
- Within the mark area, the <mark> is left-justified if <pad> is 0. If <pad> is greater than 0, the number of blanks indicated by the value of <pad> are appended to the <mark>; and the <mark indent> value is ignored. The resulting string immediately precedes the text. The mark is effectively right-justified <pad> spaces immediately to the left of text.
- Arguments <type> and <mark> interact to control the type of marking used. If <type> is 0, simple marking is performed, using the mark character(s) found in the <mark> argument. If <type> is greater than 0, automatic numbering or alphabetizing is done, and <mark> is then interpreted as the first item in the sequence to be used for numbering or alphabetizing; i.e., it is chosen from the set (1, A, a, I, i). This is summarized in the following table.

type	mark	result
0	omitted	hanging indent
0	string	string is the mark
>0	omitted	Arabic numbering
>0	one of: 1, A, a, I, i	alphabetic sequencing

Each non-zero value of <type> from one to six selects a different way of displaying the marks. The following table shows the output appearance for each value of <type>:

Value	Appearance
1	x.
2	x)
3	(x)
4	[x]
5	<x>
6	{x}

where x is the generated number or letter.

Note

The mark must not contain ordinary (paddable) spaces because alignment of items will be lost if the right margin is justified.

- The <LI space> argument gives the number of blank lines (halves of a vertical space) that should be output by each .LI macro in the list. If omitted, LI-space defaults to 1 (the value 0 can be used to obtain compact lists). If LI-space is greater than 0, *.LI* macro issues a *.ne* request for two lines just before printing the mark.
- The <LB space> argument is the number of blank lines (one half of a vertical space) to be output by .LB itself. If omitted, <LB space> defaults to 0.

There are three combinations of <LI space> and <LB space>:

- The normal case is <LI space> = 1 and <LB space> = 0, yielding one blank line before each item in the list. Such lists are usually terminated with a *.LE 1* macro to insert a blank line following the list.
- For a more compact list, <LI space> = 0 and <LB space> = 1, yielding no space between items and a blank line before and after the list. The *.LE 1* macro is used at the end of the list.
- If both <LI space> and <LB space> are zero and *.LE* is used to end the list, a list with no blank lines results.

The next topic shows how to build upon the supplied list of macros to obtain other kinds of lists.

User-defined List Structures

Note

This part is intended only for users accustomed to writing formatter macros.

If a large document requires complex list structures, it is useful to define the appearance for each list level only once instead of having to define the appearance at the beginning of each list. This promotes consistency of style in a large document. A generalized list-initialization macro might be defined in such a way that what the macro does depends on the list-nesting level in effect at the time the macro is called. For example, a macro could be constructed such that levels 1 through 5 of the list to be formatted have the following appearance:

A.

[1m

(•) bullet here

a)

\+

The following code defines a macro (*.aL*) that always begins a new list and determines the type of list according to the current list level. To understand it, remember that the number register *:g* is used by the MM list macros to determine the current list level (it is 0 if there is no currently active list). Each call to a list-initialization macro increments *:g*, and each *.LE* call decrements it.

```
\" register g is used as a local temporary to save
\" :g before it is changed below
.de aL
.nr g \\n(:g
.if\\ng=0 .AL A                    \" produces an A.
.if\\ng=1 .LB\\n(Li 0 1 4          \" produces a [1]
.if\\ng=2 .BL                      \" produces a bullet

.if\\ng=3 .LB\\n(Li 0 2 2 a        \" produces an a)

.if\\ng=4 .ML +                    \" produces a +

...
```

This macro can be used (in conjunction with *.LI* and *.LE*) instead of *.AL*, *.RL*, *.BL*, *.LB*, and *.ML*. For example, the following input:

```
.aL
.LI
First line.
.aL
.LI
Second line.
.LE
.LI
Third line.
.LE
```

when formatted, yields

A. First line.

[1] Second line.

B. Third line.

There is another approach to lists that is similar to the *.H* mechanism. List-initialization, as well as the *.LI* and the *.LE* macros, are all included in a single macro. That macro (defined as *.bL* below) requires an argument to tell it what level of item is required. It adjusts the list level by either beginning a new list or setting the list level back to a previous value. It then issues a *.LI* macro call to produce the item.

```
.de bL
.ie \\n(.$ .nr g \\$1           \"if there is an argument,
.                               \"that is the level
.el .nr g \\n(:g                \"if no argument, use current level
.if \\ng-\\n(:g>1.)D            \"**ILLEGAL SKIPPING OF LEVEL
.                               \"increasing level by more than 1
.if \\ng>\\n(:g \{.aL \\ng-1     \"if g>:g begin new list
.nr                \"and reset g to current level
.                               \"(.aL changes g)
.if \\n(:g>\\ng .LC \\ng            \"if:g>g, prune back to
.                               \"correct level
.                               \"if:g=g, stay within
.                               \"current list
.LI                             \"in all cases, get out an item
...
```

For *.bL* to work, the previous definition of the *.aL* macro must be changed to obtain the value of *g* from its argument rather than from *:g*. Invoking *.bL* without arguments causes it to stay at the current list level. The *.LC* (List Clear) macro removes list descriptions until the level is less than or equal to that of its argument. For example, the *.H* macro includes the call `.LC 0`. If text is to be resumed at the end of a list, insert the call `.LC 0` to clear out the lists completely. The example below illustrates the relatively small amount of input needed by this approach. The input text:

```
The quick brown fox jumped over the lazy dog's back.
.bL 1
First line.
.bL 2
Second line.
.bL 1
Third line.
.bL
Fourth line.
.LC 0
Fifth line.
```

when formatted, yields

The quick brown fox jumped over the lazy dog's back.

a. First line.

    [1] Second line.

b. Third line.
c. Fourth line.

Fifth line.

Memorandum and Released-Paper Style Documents

One use of MM is for the preparation of memoranda and released-paper documents (a documentation style used by some large corporations) which have special requirements for the first page and for the cover sheet. Data needed (title, author, date, case numbers, etc.) is entered in the same way for both styles; an argument to the *.MT* macro indicates which style is being used.

Sequence of Beginning Macros

Macros, if present, must be given in the following order:

```
.ND <new date>
.TL [<charging case>] [<filing case>]
one or more lines of text
.AF [<company name>]
.AU <name> [<initials>] [<loc>] [<dept>] [<text>] [<room>] [<arg>] [<arg>]
.AT [<title>] ...
.TM [<number>] ...
.AS [<arg>] [<indent>]
one or more lines of text
.AE
.NS [<arg>]
one or more lines of text
.NE
.OK [<keyword>] ...
.MT [<type>] [<addressee>]
```

The only required macros for a memorandum for file or a released-paper document are *.TL*, *.AU*, and *.MT*. All other macros (and other associated input lines) can be omitted if the features are not needed. Once *.MT* has been invoked, none of the above macros (except *.NS* and *.NE*) can be reinvoked because they are removed from the table of defined macros to save memory space.

If neither the memorandum nor released-paper document style is desired, the TL, AU, TM, AE, OK, MT, ND, and AF macros should be omitted from input text. If these macros are omitted, the first page will have only the page header followed by the body of the document.

Title

```
.TL [<charging case>] [<filling case>]
one or more lines of title text
```

Arguments to the .TL macro are the <charging case> number(s) and <filing case> number(s).

- The <charging case> argument is the case number to which time was charged for the development of the project described in the memorandum. Multiple <charging case> numbers are entered as "subarguments" by separating each from the previous with a comma and a space and enclosing the entire argument within double quotes.
- The <filing case> argument is a number under which the memorandum is to be filed. Multiple filing case numbers are entered similarly. For example:

```
.TL "12345,67890" 987654321
construction of a Table of all Even Prime numbers
```

The title of the memorandum or released-paper document follows the *.TL* macro and is processed in fill mode. The *.br* request can be used to break the title into several lines as follows:

```
.TL 12345
First Title Line
.br
\!.br
Second Title Line
```

On output, the title appears after the word "subject" in the memorandum style and is centered and bold in the released-paper document style.

If only a charging case number or only a filing case number is given, it will be separated from the title in the memorandum style by a dash and will appear on the same line as the title. If both case numbers are given and are the same, the "Charging and Filing Case" followed by the number will appear on a line following the title. If the two case numbers are dirrerent, separate lines for "Charging Case" and "File Case" will appear after the title.

Authors

```
.AU <name> [<initials>] [<loc>] [<dept>] [<ext>] [<room>] [<arg>] [<arg>]
.AT [<title>] ...
```

The *.AU* macro receives as arguments information that describes an author. If any argument contains blanks, that argument must be enclosed within double quotes. The first six arguments must appear in the order given. A separate *.AU* macro is required for each author.

The *.AT* macro is used to specify the author's title. Up to nine arguments can be given. Each will appear in the signature block for memorandum style on a separate line following the signer's name. The *.AT* must immediately follow the .AU for the given author. For example:

```
.AU"J. J. Jones" JJJ PH 9876 5432 1Z-234
.AT Director "Materials Research Laboratory"
```

In the "from" portion in the memorandum style, the author's name is followed by location and department number on one line and by room number and extension number on the next. The "x" for the extension is added automatically. Printing of the location, department number, extension number, and room number can be suppressed on the first page of a memorandum by setting register *Au* to 0; (default value for *Au* is 1). Arguments 7 through 9 of the .AU macro, if present, will follow this normal author information in the "from" portion, each on a separate line. These last three arguments can be used for organizational numbering schemes, etc. For example:

```
.AU "S.P. Lename" SPL IH 9988 7766 5H-444 9876-543210.01MF
```

The name, initials, location, and department are also used in the signature block. Author information in the "from" portion, as well as names and initials in the signature block, will appear in the same order as the *.AU* macros.

Names of authors in the released-paper style are centered below the title. Following the name of the last author, the company name and location are centered. The paragraph on memorandum types contains information regarding authors from different locations.

TM Numbers

```
.TM [<number>] ...
```

If the memorandum is a technical memorandum, the TM numbers are supplied via the *.TM* macro. Up to nine numbers can be specified. For example:

```
.TM 7654321 77777777
```

This macro call is ignored in the released-paper and external-letter styles.

Abstracts

```
.AS [<arg>] [<indent>]
text of abstract
.AE
```

If a memorandum has an abstract, the input is identified with the `.AS` (abstract start) and `.AE` (abstract end) delimiters. Abstracts are printed on page 1 of a document and/or on its cover sheet. There are three styles of cover sheet:

- Released paper
- Technical memorandum
- Memorandum for file (also used for engineer's notes, memoranda for record, etc.)

Cover sheets for released papers and technical memoranda are obtained by invoking the *.CS* macro.

In released-paper style (<type> argument of the .MT macro is 4), and in technical memorandum style, if the first argument (<arg>) of *.AS* is:

0 Abstract is printed on page 1 and on the cover sheet.

1 Abstract apppears only on the cover sheet (if any).

In memoranda-for-file style and in all other documents (other than external letters) if the <arg> of *.AS* is:

0 Abstract appears on page 1 and no cover sheet is printed.

2 Abstract appears only on the cover sheet which is produced automatically (i.e., without invoking the .CS macro).

No abstract or cover sheet can be obtained with an external-letter style (<type> argument of .MT macro is 5).

Notations such as a "copy to" list are allowed on memorandum-for-file cover sheets; *.NS* and *.NE* commands must apppear after *.AS 2* and *.AE*. Headings and displays are not permitted within an abstract.

The abstract is printed with ordinary text margins (the indentation to be used for both margins can be specified as the <indent> argument of .AS). Values that specify indentation must be unscaled, and are treated as "character positions"; i.e., as the number of ens.

Other Keywords

```
.OK [<keyword>] ...
```

Topical keywords should be specified on a technical memorandum cover sheet. Up to nine such keywords or keyword phrases can be specified as arguments to the *.OK* macro. If any keyword contains spaces, it must be enclosed within double quotes.

Memorandum Types

```
.MT [<type>] [<addressee>]
```

The *.MT* macro controls the format of the top part of the first page of a memorandum or released-paper document, and the format of the cover sheets. Allowable <type> arguments and corresponding document functions are:

Type	Document Function
" "	No memorandum type printed
0	No memorandum type printed
(omitted)	MEMORANDUM FOR FILE
1	MEMORANDUM FOR FILE
2	PROGRAMMER'S NOTES
3	ENGINEER'S NOTES
4	Released-paper style
5	External-letter style
"string"	String (enclosed in quotes)

If <type> indicates a memorandum-style document, the corresponding statement indicated under "Document Function" is printed after the last line of author information. If <type> is longer than one character, the <type> string itself is repinted as in this example:

```
.MT "Technical Note #5"
```

A simple letter is produced by calling *.MT* with a null (but not omitted) or 0 argument.

The <addressee> argument to *.MT* is the letter-addressee's name. If present, that name and the page number replace the normal page header on the second and following pages of a letter. For example:

```
.MT 1"John Jones"
```

produces

John Jones - 2

The <addressee> argument cannot be used if the <type> argument is 4 (released-paper style document).

The released-paper style is obtained by specifying

```
.MT 4 [1]
```

This results in a centered, bold title followed by centered names of authors. The location of the last author is used as the location following "Bell Laboratories" (unless the .AF macro has been edited to specify a different company). If the optional second argument to .MT 4 is given, the name of each author is followed by the respective company name and location. Information necessary for the memorandum style document but not for the released-paper style document is ignored.

If the released-paper style document is utilized, most company location codes are defined as strings that are the addresses of the corresponding company locations. These codes are needed only until the *.MT* macro is invoked. Thus, following the *.MT* macro, the user can reuse these string names, In addition, the macros for the end of a memorandum and their associated lines of input are ignored when the released-paper style is specified.

Company locations in "as-shipped" MM refer to Bell Telephone Laboratories. To use other company locations, edit the *.MT* macro, or include author affiliations in the released-paper style by specifying the appropriate *.AF* macro, defining a string (with a 2-character name such as ZZ) for the address before each *.AU*. For example:

```
.TL
A Learned Treatise
.AF "Getem Inc."
.ds ZZ "22 Maple Avenue, Sometown 09999"
.AU "F. Swatter" "" ZZ
.AF "Bell Laboratories"
.AU "Sam P. Lename" "" CB
.MT 4 1
```

In the external-letter style document (.MT 5), only the title (without the word "subject:") and the date are printed in the upper left and right corners, respectively, on the first page. It is expected that preprinted stationery will be used with the company logo and address of the author.

Date Changes

```
.ND <new date>
```

The *.ND* macro alters the value of the string DT, which is initially set to produce the current date. If the argument contains spaces, it must be enclosed within double quotes.

Alternate First-Page Format

```
.AF [<company name>]
```

An alternate first-page format can be specified with the *.AF* macro. The words "subject", "date", and "from" (in the memorandum style) are omitted and an alternate company name is used.

If an argument is given, it replaces "Bell Laboratories" without affecting other headings. If the argument is null, "Bell Laboratories" is suppressed; and extra blank lines are inserted to allow room for stamping the document with a logo or company-name stamp.

.AF with no argument suppresses "Bell Laboratories" and the "Subject/Date/From" headings, thus allowing output on preprinted stationery. The use of *.AF* with no arguments is equivalent to the use of `-rA1`, except that the latter must be used if it is necessary to change the line length and/or page offset (which default to 5.8i and 1i, respectively, for preprinted forms). The command line options `-rOK` and `-rWK` do not affect *.AF*. The only *.AF* use appropriate for *troff* is when specifying a replacement for "Bell Laboratories".

The command line option `-rEn` controls the font of the "Subject/Date/From" block.

Example

Input text for a typical document could resemble the following:

```
.TL
MM/*(EMMemorandum Macros
.AU "H. R. Douglas" DWS PY
.Au "E. S. Mahoney" JRM PY
.AU "W. C. Archer (January 1984 Revision)" ECP PY
.AU "R. E. Taylor (June 1984 Revision)" NWS PY
.MT 4
```

End-of-Memorandum Macros

At the end of a memorandum document (but not a released-paper document), signatures of authors and a list of notations can be requested. The following macros and their input are ignored if the released-paper style document is selected.

Signature Block

```
.FC [<closing>]
.SG [<arg>] [1]
```

The *.FC* macro prints "Yours very truly," as a formal closing, if no argument is used. It must be given before the *.SG* macro. A different closing can be specified as an argument to *.FC*.

.SG prints the author's name(s) after the formal closing, if any. Each name begins at the center of the page. Three blank lines are left above each name for the signature.

- If no arguments are given, the line of reference data (location code, department number, author's initials, and typist's initials – all separated by hyphens) are omitted.
- A non-null first argument is treated as the typist's initials and is appended to the reference data.
- A null first argument prints reference data without the typist's initials or the preceding hyphen.
- If there are several authors and the second argument is given, reference data is placed on the same line as the first author.

Reference data contains only the location and department number of the first author. Thus, if there are authors from different departments and/or from different locations, the reference data should be supplied manually after invocation (without arguments) of the *.SG* macro. For example:

```
.SG
.rs
.sp -1v
PY/MH-9876/5432-JJJ/SPL-cen
```

"Copy to" and Other Notations

```
.NS [<arg>]
zero or more notation lines
.NE
```

Many types of notation (such as a list of attachments or "Copy to" lists) can follow signature and reference data. Various notations are obtained through the *.NS* macro, which provides for proper spacing and for breaking notations across pages, if necessary.

Codes for <arg> and the corresponding printed notations are:

<arg>	**Notations**
none	Copy to
""	Copy to
0	Copy to
1	Copy to
2	Copy (with att.) to
3	Att.
4	Atts.
5	Enc.
6	Encs.
7	Under Separate Cover
8	Letter to
9	Memorandum to
"string"	Copy (string) to

If <arg> consists of more than one character, it is placed whithin parentheses between the words "Copy" and "to". For example:

```
.NS "with att. 1 only"
```

generates

```
Copy (with att. 1 only) to
```

as the notation. More than one notation can be specified before the *.NE* macro because a *.NS* macro terminates the preceding notation, if any. For example:

```
.NS 4
Attachment 1-List of register names
Attachment 2-List of string and macro names
.NS 1
J. J. Jones
.NS 2
S. P. Lename
G. H. Hurtz
.NE
```

would be formatted as

```
Atts.
Attachment 1-List of register names
Attachment 2-List of string and macro names

Copy (with att.) to
J. J. Jones

Copy (without att.) to
S. P. Lename
G. H. Hurtz
```

.NS and *.NE* can also be used at the beginning after *.AS 2* and *.AE* to place the notation list on the memorandum-for-file cover sheet. If notations are given at the beginnning without *.AS 2*, they are saved for output at the end of the document.

Approval Signature Line

```
.AV <approver's name>
```

The *.AV* macro can be used after the last notation block to automatically generate a line with spaces for the approval signature and date. For example:

```
.AV "Jane Doe"
```

produces

APPROVED:

_____ _____

Jane Doe Date

One-Page Letter

At times, you may want more space on the page, thus forcing the signature or items within notations to the bottom of the page so that the letter or memo is only one page in length. This can be accomplished by increasing the page length with the `-rLn` option, e.g., `-rL90`. This has the effect of making the formatter believe that the page is 90 lines long and therefore providing more space than usual for placing the signature or notations.

Note

This works **only** for one-page letters and memos.

Displays

Displays are blocks of text that are to be kept together on a page and not split across pages. They are processed in an environment that is different from the body of the text (see the *.ev* request). The MM package provides two styles of displays: a static (.DS) style and a floating (.DF) style.

- In the static style, the display appears in the same relative position in the output text as it does in the input text. This may result in extra white space at the bottom of the page if the display is too long to fit in the remaining page space.
- In the floating style, the display "floats" through the input text to the top of the next page if there is not enough space on the current page. Thus, input text that follows a floating display may precede it in the output text. A queue of floating displays is maintained so that their relative order of appearance in the text is not disturbed.

By default, a display is processed in no-fill mode with single spacing and is not indented from the existing margins. The user can specify indentation and centering, as well as fill-mode processing.

Note

Displays and footnotes cannot be nested in any combination. Although lists and paragraphs are permitted, no headings (.H or .HU) can be used within displays or footnotes.

Static Displays

```
.DS [<format>] [<fill>] [<rindent>]
one or more lines of text
.DE
```

A static display is started by the *.DS* macro and terminated by the *.DE* macro. With no arguments, *.DS* accepts lines of text exactly as typed (no-fill mode), and will not indent lines from the prevailing left margin indentation or from the right margin.

- The <format> argument is an integer or letter used to control the left-margin indentation and centering with the following meanings:

<format>	Meaning
" "	no indent
0 or L	no indent
1 or I	indent by standard amount
2 or C	center each line
3 or CB	center as a block
(omitted)	no indent

- The <fill> argument is an integer or letter and can have the following meanings:

<fill>	Meaning
" "	no-fill mode
0 or N	no-fill mode
1 or F	fill mode
(omitted)	no-fill mode

- The <rindent> argument is the number of characters that the line length should be decreased, i.e., an indentation from the right margin. This number must be unscaled for *nroff*, and is treated as ens. It can be scaled when using *troff*, and defaults to ems.

The standard amount of static display indentation is taken from the *Si* register, a default value of five spaces. Thus, text of an indented display aligns with the first line of indented paragraphs whose indent is contained in the *Pi* register. Even though their initial values are the same (default), these two registers are independent.

The display format argument value 3 (or CB) horizontally centers the entire display as a block (as opposed to *.DS 2* and *.DF 2* which center each line individually). All collected lines are left-justified and the display is centered (based on the width of the longest line). With *eqn/neqn*, this format must be used in order for the "mark" and "lineup" features to work when dealing with centered equations.

By default, a blank line (one half of a vertical space) is placed before and after static and floating displays. These blank lines before and after static displays can be inhibited by setting register *Ds* to 0.

The following example shows how to use all three arguments for static displays. This block of text will be indented five spaces from the left margin, filled, and indented five spaces from the right margin (i.e., centered). The input text:

```
.DS I F 5
"We the people of the United States,
in order to form a more perfect union,
establish justice, ensure domestic tranquillity,
provide for the common defense,
and secure the blessings of liberty to
ourselves and our posterity,
do ordain and establish this Constitution for the
United States of America."
.DE
```

produces:

> "We the people of the United States, in order to form a more perfect union, establish justice, ensure domestic tranquillity, provide for the common defense, and secure the blessings of liberty to ourselves and our posterity, do ordain and establish this Constitution for the United States of America."

Floating Displays

```
.DF [<format>] [<fill>] [<rindent>]
one or more lines of text
.DE
```

A floating display is started by the *.DF* macro and terminated by the *.DE* macro.

Arguments have the same meanings as static displays described above, except that indent, no indent, and centering are calculated with respect to the initial left margin. This is because prevailing indent may change between the time the formatter first reads the floating display and when the display is printed. One blank line (one half of a vertical space) occurs before and after a floating display.

You can exercise precise control over the output positioning of floating displays by using two number registers, *De* and *Df*. When a floating display is encountered by *nroff* or *troff*, it is processed and placed in a queue of displays waiting for output. Displays are removed from the queue and printed in the order entered, which is the same order that they appeared in the input file. If a new floating display is encountered and the display queue is empty, the new display is a candidate for immediate output on the current page. Immediate output is governed by the size of the display and the *Df* register code setting. The *De* register code controls whether any additional text will follow the display if it is printed on the current page.

As long as the display queue contains one or more displays, new displays are automatically placed there, rather than being output. When a new page is started (or the top of the second column when in 2-column mode), the next display from the queue becomes a candidate for output if the Df register code has specified "top-of-page" output. When display-output processing is complete, the display is removed from the queue.

When the end of a section (using section-page numbering) or the end of a document is reached, all remaining displays are automatically output and removed from the queue. This occurs before a `.SG`, `.CS`, or `.TC` macro is processed.

A display will fit on the current page if there is enough room to contain the entire display or if the display is longer than one page in length and less than half of the current page has been used. A wide (full-page width) display will not fit in the second column of a 2-column document.

The *De* and *Df* number register code settings and actions are as follows:

_De_ Register

Code	Action
0	No special action occurs (also the default condition).
1	Each floating display is followed by a page eject such that only one floating display appears on a given page. No other text follows the display.

Any *De* <code> value other than 0 is interpreted the same as <code> = 1.

Code	Action
0	Floating displays are not output until end of section (if section-page numbering is enabled) or end of document (all other cases).
1	Output new floating display on current page if there is space; otherwise, hold it until end of section or document.
2	Output exactly one floating display from queue, beginning at the top of a new page (or column when in 2-column mode).
3	Output one floating display on current page (or column) if there is space; otherwise, begin output at the top of a new page or column.
4	Output as many displays as will fit (at least one) starting at the top of a new page or column.
5	Output a new floating display on the current page if there is room (also the default condidition). Output as many displays (at least one) as will fit on the page starting at the top of a new page or column.

Note

If *Df* <code> = 4 or 5, and *De* <code> = 1, each display is always followed by a page eject causing a new top of page to be reached. This causes the output of at least one more display.

Also, for any *Df* <code> greater than 5, the action performed is the same as when <code> = 5.

Tables

```
.TS [H]
global options;
column descriptors.
title lines
[.TH [N]]
data within the table.
.TE
```

You can use the `.TS` (table start) and `.TE` (table end) macros to access the *tbl*(1) program. These macros delimit text to be examined by *tbl* and set proper spacing around the table. The display function and the *tbl* delimiting function are independent. To ensure that blocks containing any mixture of tables, equations, filled test, unfilled text, and caption lines are kept together, the .TS/.TE block should be enclosed within a display (.DS/.DE). Floating tables can be enclosed inside floating displays (.DF/.DE).

You can also use `.TS` and `.TE` to process tables that extend over several pages.

If a table heading is needed for each page of a multipage table, the `H` argument should be included with the `.TS` macro call as above. Type the table title after the options and format information, using as many lines as you need, then follow the title with the `.TH` macro. The `.TH` macro must be included when `.TS H` is used for multipage tables (this restriction is imposed by MM, not *tbl*).

You may need to build long tables from smaller .TS H/.TE segments, requiring that table headers be suppressed for all tables except the first table on a page. To do this, use the .TH (table header) macro with the argument N. The N suppresses all table headers **except** the first header appearing on a given page. For example:

```
.TS H
global options;
column descriptors.
Title lines
.TH
data
.TE
.TS H
global options;
column descriptors.
Title lines
.TH N
data
.TE
```

outputs a table heading at the top of the first table segment and suppresses the heading at the top of the second segment when both appear on the same page. However, if the table continues at the top of the next page, a new header is output at the top of each page until the end of the table is reached.

This feature is helpful when a single table must be broken into segments because of table complexity (such as too many blocks of filled text). If each segment had its own .TS H/.TH sequence, it would have its own header. However, if each table segment after the first uses .TS H/.TH N, the table header appears only at the beginning of the table and the top of each new page or column that the table continues onto.

For the *nroff* formatter, the −e option [-E for *mm*(1)] can be used for terminals (such as the DASI 450) that are capable of finer printing resolution. This produces better alignment of features such as lines forming the corner of a box. The −e option does not affect *col*(1).

Equations

```
.DS
.EQ [label]
equation(s)
.EN
.DE
```

Mathematical typesetting programs *eqn*(1) and *neqn* expect to use the *.EQ* (equation start) and *.EN* (equation end) macros as delimiters in the same way that *tbl*(1) uses .TS and .TE; however, .EQ and .EN must occur inside a .DS/.DE pair. There is an exception to this rule − if .EQ and .EN are used to specify only the delimiters for in-line equations or to specify *eqn/neqn* defines, the .DS and .DE macros must not be used. Otherwise, extra blank lines will appear in the output.

The *.EQ* macro takes an argument that is used as a label for the equation.

The default label location is at the right margin in the "vertical center" of the general equation. Set the *Eq* register to 1 to move the label position to the left margin.

The equation is centered for centered displays. Otherwise, the equation is adjusted to the margin opposite the label.

Figure, Table, Equation, and Exhibit Titles

```
.FG [<title>] [<override>] [<flag>]
.TB [<title>] [<override>] [<flag>]
.EC [<title>] [<override>] [<flag>]
.EX [<title>] [<override>] [<flag>]
```

The `.FS` (figure title), `.TB` (table title), `.EC` (equation caption), and `.EX` (exhibit caption) macros are normally used inside .DS/.DE pairs to automatically number and title figures, tables, and equations. These macros use registers *Fg*, *Tb*, *Ec*, and *Ex*, respectively. For example:

```
.FG "This is a Figure Title"
```

yields:

Figure 1. This is a Figure Title

The *.TB* macro replaces "Figure" with "TABLE". The *.EC* macro replaces "Figure" with "Equation", and *.EX* replaces "Figure" with "Exhibit". The output title is centered if it can fit on a single line. Otherwise, the first line is centered, and the remaining title lines are lined up with the first character of the first line Number format can be changed by using the *.af* formatter request. Caption format can be changed from

Figure 1. Title

to

Figure 1–Title

by setting the *Of* register to 1.

The <override> argument is used to modify normal numbering. If <flag> is omitted or 0, <override> is used as a prefix to the number; if <flag> = 1, <override> is used as a suffix; and if <flag> = 2, override replaces the number. If `-rN5` is given, "section-figure" numbering is set automatically and any user-specified <override> string is ignored.

As a matter of formatting style, table headings are usually placed above the text of tables, while figure, equation, and exhibit titles are usually placed below corresponding figures and equations.

List of Figures, Tables, Equations, and Exhibits

A list of figures, tables, exhibits, and equations are printed following the table of contents if number registers *Lf*, *Lt*, *Lx*, and *Le* are all set to 1. Default value for *Lf*, *Lt*, and *Lx* is 1; default for *Le* is 0.

Titles of these lists can be changed by redefining the following strings (shown here with their default values):

```
.ds Lf LIST OF FIGURES
.ds Lt LIST OF TABLES
.ds Lx LIST OF EXHIBITS
.ds Le LIST OF EQUATIONS
```

Footnotes

There are two macros (.FS and .FE) that delimit footnote text, a string that automatically numbers footnotes, and a macro (.FD) that specifies the style of footnote text. Footnotes are processed in an environment different from that of the body of text (refer to *.ev* request).

Automatic Footnote Numbering

Footnotes can be automatically numbered by typing the three characters `\*F` (i.e., invoking the string F) immediately after the text to be footnoted without any intervening spaces. This places the next sequential footnote number (in a smaller point size) a half-line above the text to be footnoted.

Delimiting Footnote Text

```
.FS [<label>]
one or more lines of footnote text
.FE
```

There are two macros that delimit the text of each footnote. The *.FS* (footnote start) macro marks the beginning of footnote text, and the *.FE* (footnote end) macro marks the end. The <label> on *.FS*, if present, is used to mark footnote text. Otherwise, the number retrieved from the string *F* is used.

Automatically numbered and user-labeled footnotes can be intermixed. If a footnote is labeled (.FS <label>), the text to be footnoted must be followed by <label>, rather than by `\*F`. Text between `.FS` and `.FE` is processed in fill mode. Another `.FS`, a `.DS`, or a `.DF` are not permitted between `.FS` and `.FE` macros. If footnotes are required in the title, abstract, or table, only labeled footnotes will appear properly. Everywhere else automatically numbered footnotes work correctly. For example:

Automatically numbered footnote:

```
This is the line containing the word\*F
.FS
This is the text of the footnote.
.FE
to be footnoted.
```

Labelled footnote:

```
This is a labeled*
.FS *
The footnote is labeled with an asterisk.
.FE
footnote.
```

Footnote text (enclosed within the .FS/.FE pair) should immediately follow the word to be footnoted in the input text, so the "*F" or label occurs at the end of a line of input and the next line is the .FS macro call. It is also good practice to append an unpaddable space to "^*F" or label when they follow an end-of-sentence punctuation mark (i.e., period, question mark, or exclamation point).

The example at the end of this tutorial illustrates several footnote styles as well as numbered and labelled footnotes.

Footnote Text Format Style

`.FD [<arg>] [1]`

You can control formatting style within the footnote text by specifying text hyphenation, right-margin justification, and text indentation, as well as left or right justification of the label when text indenting is used. The .FD macro is invoked to select the appropriate style.

The first argument is a number from the left column of the followng table. Formatting style for each number is indicated in the remaining four columns. Further explanation of the first two of these columns is given in the definition of the *.ad*, *.hy*, *.na*, and *.nh* (adjust, hyphenation, no adjust, and no hyphenation, respectively) requests in the *nroff* tutorial.

<arg>	HYPHENATION	ADJUST	TEXT INDENT	LABEL JUSTIFICATION
0	.nh	.ad	yes	left
1	.hy	.ad	yes	left
2	.nh	.na	yes	left
3	.hy	.na	yes	left
4	.nh	.ad	no	left
5	.hy	.ad	no	left
6	.nh	.na	no	left
7	.hy	.na	no	left
8	.nh	.ad	yes	right
9	.hy	.ad	yes	right
10	.nh	.na	yes	right
11	.hy	.na	yes	right

If the argument to *.FD* is greater the 11, it is handled as if `.FD 0` were specified. If the first argument is omitted or null, the effect is equivalent to `.FD 10`" in *nroff* and `.FD 0` in *troff*. These are also the respective initial default values.

If the second argument is specified, automatically-numbered footnotes begin again with 1 whenever a first-level heading is encountered. This is most useful with the "section-page" page numbering scheme. As an example, the input line

```
.FD " " 1
```

maintains the default formatting style and causes footnotes to be numbered afresh after each first-level heading in a document.

Hyphenation across pages is inhibited by MM except for long footnotes that continue to the following page. If hyphenation is permitted, it is possible for the last word on the last line on the current page footnote to be hyphenated. You can suppress end-of-page hyphenation by specifying an even .FD argument.

Footnotes are separated from the body to the text by a short line rule. Footnotes that continue to the next page are separated from the body of the text by a full-width rule. *Troff* sets footnotes in a type size two points smaller than the text point size.

Spacing Between Footnote Entries

Normally, one blank line (a 3-point vertical space) separates footnotes when more than one occurs on a page. To change this spacing, number register *Fs* is set to the desired value. For example:

```
.nr Fs 2
```

produces two blank lines (a 6-point vertical space) between footnotes.

Page Headers and Footers

Text printed at the top of each page is called a **page header**. Text printed at the bottom of each page is called a **page footer**. Up to three lines of text can be associated with the header. The first line is printed on every page, the second on even pages; the third on odd pages only. Thus each printed page header can have up to two lines of text: the line that occurs at the top of every page, and the line for even- or odd-numbered pages. The same arrangement applies to page footers.

This section describes the default appearance of page headers and footers and explains how to change them. The terms **header** and **footer** (not qualified by even or odd) is used to mean the page-header or page-footer line that occurs on every page.

Default Headers and Footers

Each page has a default page header consisting of a centered page number. There is no default footer or default even/odd header/footer except in section-page numbering schemes explained later in this section.

In a memorandum or a released-paper style document, the page header on the first page is automatically suppressed unless a break occurs before the .MT macro is called. Macros and text in the following categories do not cause a break, and can precede the memorandum types (.MT) macro in an input text file.

- Memorandum and released-paper style document macros (*.TL*, *.AU*, *.AT*, *.TM*, *.AS*, *.AE*, *.OK*, *.ND*, *.AF*, *.NS*, and *.NE*)
- Page header and footer macros (*.PH*, *.EH*, *.OH*, *.PF*, *.EF*, and *.OF*)
- The *.nr* and *.ds* requests.

Header and Footer Macros

For header and footer macros (*.PH*, *.EH*, *.OH*, *.PF*, *.EF*, and *.OF*), the argument [<arg>] is of the following form:

```
" '<left part>'<center part>'<right part>' "
```

If it is inconvenient to use an apostrophe (') as the delimiter because it occurs within one of the parts, it can be replaced uniformly by any other character. In formatted output, the parts are left-justified, centered, and right-justified, respectively.

Page Header

```
.PH [<arg>]
```

The *.PH* macro specifies the header that is to appear at the top of every page. The initial value is the default centered page number enclosed by hyphens. The page number contained in the *P* register is an Arabic number. The format of the number can be changed by using the *.af* macro request.

If "debug mode" is set by including the `-rD1` flag on the command line, additional information printed at the top left of each page is included in the default header. This consists of the Source Code Control System (SCCS) release and level of MM (thus identifying the current version) followed by the current line number within the current input file.

Even-Page Header

```
.EH [<arg>]
```

The *.EH* macro supplies a line to be printed at the top of each even-numbered page immediately following the header. Initial value is a blank line.

Odd-Page Header

```
.OH [<arg>]
```

The *.OH* macro is identical to *.EH* except that it applies to odd-numbered pages.

Page Footer

```
.PF [<arg>]
```

The *.PF* macro specifies the line that is to appear at the bottom of each page. Its initial value is a blank line. If the `-rCn` flag is specified on the command line, the type of copy follows the footer on a separate line. In particular, if `-rC3` or `-rC4` (DRAFT) is specified, the footer is initialized to contain the date instead of being a blank line.

Even-Page Footer

```
.EF [<arg>]
```

The *.EF* macro supplies a line to be printed at the bottom of each even-numbered page immediately preceding the footer. Initial value is a blank line.

Odd-Page Footer

```
.OF [<arg>]
```

The *.OF* macro is identical to *.EF* except that it applies to odd-numbered pages.

First-Page Footer
The default first page footer is a blank line. If you specify `.PF` and/or `.OF` before the end of the first page of the document, the specified lines will be printed at the bottom of the first page.

The header (whatever its contents) replaces the footer (on the first page only) if the `-rN1` flag is specified on the command line.

Default Header and Footer With Section-Page Numbering

Pages can be numbered sequentially within sections by "section-number page-number". To obtain this numbering style, `-rN3` or `-rN5` is specified on the command line. In this case, the default footer is a centered "section-page" number, and the default page header is blank.

Strings and Registers in Header and Footer Macros

String and register names can be placed in arguments to header and footer macros. If the value of the string or register is to be computed when the respective header or footer is printed, invocation must be escaped by four backslashes because string or register invocation is processed three times:

1. As the argument to the header or footer macro,
2. In a formatting request within the header or footer macro,
3. In a *.tl* request during header or footer processing.

For example, page number register *P* must be escaped with four backslashes in order to specify a header in which the page number is to be printed at the right margin. For example:

```
.PH "'''Page\\\\nP'"
```

creates a right-justified header containing the word "Page" followed by the page number. Similarly, to specify a footer with the "section-page" style, specify (see Numbered Headings Marking Styles topic for explanation of *H1*):

```
.PF"'''-\\\\n(H1-\\\\nP'"
```

If you arrange for string *a]* to contain the current section heading that is to be printed at the **bottom** of each page, the *.PF* macro call would be

```
.PF"''\\\\*(a]''"
```

If only one or two backslashes were used, the footer would print a constant value for *a]*, namely, its value when `.PF` appeared in the input text.

Header and Footer Example

The following sequence specifies blank lines for header and footer lines, page numbers on the outside margin of each page (i.e., top left margin of even pages and top right margin of odd pages), and "Revision 3" on the top inside margin of each page (nothing is specified for the center):

```
.PH ""
.PF ""
.EH "'\\\\nP''Revision3'"
.OH"'Revision3"\\\\nP'"
```

Generalized Top-of-Page Processing

Note

This part is intended only for users accustomed to writing formatter macros.

During header processing, MM invokes two user-definable macros:

- The *.TP* (top of page) macro is invoked in the environment (refer to *.ev* request) of the header.
- *.PX* is a page header user-exit macro that is invoked (without arguments) when the normal environment has been restored and the "no-space" mode is already in effect.

The default definition of .TP (after the first page of a document) is:

```
.de TP
.sp 3
.tl \\*(}t
.if e 'tl\\*(}e
.if o 'tl\\*(}o
.sp 2
...
```

String *}t* contains the header, string *}e* contains the even-page header, and string *}o* contains the odd-page header as defined by the *.PH*, *.EH*, *.EH*, and *.OH* macros, respectively. To obtain specialized page titles, redefine the *.TP* macro to produce the desired header processing. Formatting within *.TP* is processed in an environment different from that of the body. For example, to obtain a page header that includes three centered lines of data, i.e., document number, issue date, and revision date, *.TP* could be defined as follows:

```
.de TP
.sp
.ce 3
777-888-999
Iss. 2, AUG 1977
Rev. 7, SEP 1977
.sp
...
```

The *.PX* macro can be used to provide text that is to appear at the top of each page after the normal header. Tab stops can be included to align it with columns of text in the body of the document.

Generalized Bottom-of-Page Processing

```
.BS
zero or more lines of text
.BE
```

Lines of text that are specified between the *.BS* (bottom-block start) and *.BE* (bottom-block end) macros are printed at the bottom of each page after the footnotes (if any) but before the page footer. This block of text is removed by specifying an empty block, i.e.:

```
.BS
.BE
```

The bottom block appears on the table of contents, text pages, and the memorandum-for-file cover sheet, but it is not printed on technical memorandum or released-paper cover sheets.

Top and Bottom (Vertical) Margins

```
.VM [<top>] [<bottom>]
```

The *.VM* (vertical margin) macro is used to specify additional space at the top and bottom of the page. This space precedes the page header and follows the page footer. *.VM* takes two unscaled arguments that are treated as *v*'s. For example:

```
.VM 10 15
```

adds 10 blank lines to the default top-of-page margin and 15 blank lines to the default bottom-of-page margin. Both arguments must be positive (default spacing at the top of the page can be decreased by redefining *.TP*).

Proprietary Marking

```
.PM [<code>]
```

The *.PM* (proprietary marking) macro appends a PRIVATE, NOTICE, BELL LABORATORIES PROPRIETARY, or BELL LABORATORIES RESTRICTED disclaimer to the page footer. Allowable <code> arguments include:

<code>	Disclaimer
none	turn off previous disclaimer, if any
P	PRIVATE
N	NOTICE
BP	BELL LABORATORIES PROPRIETARY
BR	BELL LABORATORIES RESTRICTED

You can alter the disclaimer options by editing the *.PM* macro. To alternate disclaimers, use the .BS/.BE macro pair.

Private Documents

```
.nr Pv <value>
```

The word "PRIVATE" can be printed, centered, and underlined on the second line of a document (preceding the page header). This is done by setting the *Pv* register value:

<value>	Meaning
0	do not print PRIVATE (default)
1	PRIVATE on first page only
2	PRIVATE on all pages

If <value> = 2, the user-definable *.TP* macro cannot be used because the *.TP* macro is used by MM to print "PRIVATE" on all pages except the first page of a memorandum on which *.TP* is not invoked.

Table of Contents and Cover Sheet

The table of contents and the cover sheet for a document are produced by invoking the *.TC* and *.CS* macros, respectively.

(**Note:** This section refers to cover sheets for technical memoranda and released-papers only. The mechanism for producing a memorandum-for-file cover sheet was discussed earlier.)

These macros normally appear once at the end of the document, after the Signature Block and Notations macros, and can occur in either order.

The table of contents is produced at the end of the document because the entire document must be processed before the table of contents can be generated. Similarly, the cover sheet is produced at the end because it may not be wanted.

Table of Contents

`.TC` [<slevel>] [<spacing>] [<tlevel>] [<tab>] [<head1>] [<head2>] [<head3>] [<head4>] [<head5>]

The *.TC* macro generates a table of contents containing heading levels that were saved for the table of contents as determined by the value of the *CI* register. Arguments to *.TC* control: spacing before each entry, placement of associated page numbers, and any additional text preceding the word "CONTENTS" on the first page of the table of contents.

Spacing before each entry is controlled by <slevel> and <spacing>. Headings whose <level> is less than or equal to <slevel> will have spacing blank lines (halves of a vertical space) before them. Both <slevel> and <spacing> default to 1. This means that first-level headings are preceded by one blank line (one-half of a vertical space). The <slevel> argument does not control what heading levels have been saved (saving headings is the function of the *CI* register).

<Tlevel> and <tab> control placement of the associated page number for each heading. Page numbers can be justified at the right margin with either blanks or dots (called leaders) separating the heading text from the page number, or the page numbers can follow the heading text.

For headings whose <level> is less than or equal to <tlevel> (default 2), page numbers are justified at the right margin. In this case, the value of <tab> determines the character used to separate heading text from page number. If tab is 0 (default value), dots (i.e., leaders) are used. If tab is greater than 0, spaces are used.

For headings whose level is greater than <tlevel>, page numbers are separated from heading text by two spaces (i.e., page numbers are "ragged right", not right justified).

Additional arguments (<head1> . . . <head5>) are horizontally centered on the page, and precede the table of contents.

If the *.TC* macro is invoked with not more than four arguments, either the user-exit macro *.TX* is invoked (without arguments) before the word "CONTENTS" is printed or the user-exit macro *.TY* is invoked and the word "CONTENTS" is not printed.

By defining *.TX* or *.TY* and invoking *.TC* with four or fewer arguments, you can specify what needs to be done at the top of the first page of the table of contents. For example:

```
.de TX
.ce 2
Special Application
Message Transmission
.sp 2
.in +10n
Approved: \l'3i'
.in
.sp
...
.TC
```

yields the following output when the file is formatted:

Special Application
Message Transmission

Approved _____

CONTENTS

.
.
.

If the *.TX* macro was defined at *.TY*, the word "CONTENTS" is suppressed. Defining *.TY* as an empty macro suppresses "CONTENTS" with no replacement:

```
.de TY
...
```

By default, first-level headings are left-justified in the table of contents. Subsequent levels are then aligned with the text of headings at the preceding level. These indentations can be changed by defining the *Ci* string which takes a maximum of seven arguments corresponding to the heading levels. It must be given at least as many arguments as are set by the *Cl* register. Arguments must be scaled. For example, if *Cl* = 5:

```
.ds Ci .25i .5i .75i 1i 1i
```

or

```
.ds Ci 0 2n 4n 6n 8n
```

Two other registers are available to modify the format of the table of contents: *Oc* and *Cp*. Default page numbering for table of contents is lowercase Roman numeral. If *Oc* is set to 1, the *.TC* macro does not print any page number but instead resets the *P* register to 1. It is the user's responsibility to provide an appropriate page footer to specify page number placement. The same *.PF* (page footer) macro used in the document body is usually adequate.

The list of figures, tables, and such is produced separately unless *Cp* is set to 1, causing these lists to appear on the same page as the table of contents.

Cover Sheet

```
.CS [<pages>] [<other>] [<total>] [<figs>] [<tbls>] [<refs>]
```

The *.CS* macro generates a cover sheet in either the released-paper or technical-memorandum style (cover sheets were discussed in released-paper section). All other information for the cover sheet is obtained from data given before the `.MT` macro call. If the technical memorandum style is used, the *.CS* macro generates the "Cover Sheet for Technical Memorandum". Data appearing in the lower left corner of the technical memorandum cover sheet (counts of: pages of text, other pages, total pages, figures, tables, and references) is generated automatically (0 is used for "other pages"). These values can be changed by supplying the corresponding arguments to the *.CS* macro. If the released-paper style is used, all arguments to *.CS* are ignored.

References

There are two macros (.RS and .RF) that delimit reference text, a string that automatically numbers the subsequent references, and an optional macro (.RP) that produces references pages within the document.

Automatic Numbering of References

Automatically numbered references can be obtained by typing `\*(Rf` (invoking the string *Rf*) immediately after the text to be referenced. This places the next sequential reference number (in a smaller point size) enclosed in brackets one-half line above the text to be referenced. Reference count is kept in the *Rf* number register.

Delimiting Reference Text

```
.RS [<string name>]
.RF
```

The *.RS* and *.RF* macros are used to delimit text of each reference as shown below:

```
A line of text to be referenced.\*(Rf
.RS
reference text
.RF
```

Subsequent References

The .RS macro takes one argument, a string-name. For example:

```
.RS aA
reference text
.RF
```

The string *aA* is assigned the current reference number. This string can be used later in the document as the string call `\*(aA` to reference text that must be labeled with a prior reference number. The reference, as output, is enclosed in brackets one-half line above the text to be referenced. No `.RS`/`.RF` pair is needed for subsequent references.

Reference Page

`.RP` [<arg1>] [<arg2>]

A reference page with the default title, "References", is generated automatically at the end of the document (before table of contents and cover sheet) and is listed in the table of contents. This page contains the reference items (i.e., reference text enclosed within .RS/.RF pairs). Reference items are separated by a space (one-half of a vertical space) unless the *Ls* register is set to 0 to suppress this spacing. You can change the reference page title by defining the *Rp* string.

```
.ds Rp "New Title"
```

The *.RP* (reference page) macro can be used to produce reference pages anywhere else within a document (such as after each major section). *.RP* is not needed to produce a separate reference page with default spacings at the end of the document.

The two *.RP* macro arguments are used to control resetting of reference numbering and page skipping.

<**argl**>	**Meaning**
0	reset reference counter (default)
1	do not reset reference counter

<**arg2**>	**Meaning**
0	put on separate page (default)
1	do not cause a following .SK
2	do not cause a preceding .SK
3	no .SK before or after

If no `.SK` is issued by *.RP*, a single blank line separates the references from the following/preceding text. You may want to adjust spacing. For example, to produce references at the end of each major section:

```
.sp 3
.RP 1 2
.H 1"Next Section"
```

Miscellaneous Features

Bold, Italic, and Roman Fonts

```
.B [<bold arg>] [<previous-font arg>] . . .
.I [<italic arg>] [<previous-font arg>] . . .
.R
```

When called without arguments, *.B* changes the font to bold; *.I* to underlining (italic). This condition continues until *.R* occurs, causing the Roman font to be restored. Thus:

```
.I
here is some text.
.R
```

produces underlined text from *nroff* and italic text from *troff*(1)

If the *.B* or *.I* macro is called with one argument, that argument is printed in the appropriate font (*.I* is underlined by *nroff*), then the previous font is restored after argument output is complete. If two or more arguments (maximum six) are included with a *.B* or *.I* macro call, the second argument is concatenated to the first with no intervening space (1/12 space if the first font is italic) and printed in the previous font. Remaining pairs of arguments are similarly alternated. For example:

```
.I italic "text" right -justified
```

produces

italic text *right*-justified

The *.B* and *.I* macros alternate the specified font with the one that was prevailing when the macro was invoked. To alternate specific pairs of fonts, use one of the following macros:

.IB .BI .IR .RI .RB .BR

Each macro takes a maximum of six arguments and alternates arguments between the specified fonts, in the order specified.

If your output terminal cannot underline, insert the following at the beginning of the document to eliminate all underlining:

```
.rm ul
.rm cu
```

Note that font changes in headings are handled separately.

Right Margin Justification

```
.SA [<arg>]
```

The *.SA* macro sets right-margin justification for main-body text. Two justification flags are used: **current** and **default**. The `.SA 0` call sets both flags to **no justification** (it acts like the *.na* request). The `.SA 1` call sets both flags to cause both right- and left-justification (the same as the *.ad* request. However, calling *.SA* without an argument causes the current flag to be copied from the default flag, thus performing either a *.na* or *.ad*, depending on the default. Initially, both flags are set for no justification in *nroff* and for justification in *troff*.

In general, the no-adjust request (*.na*) can be used to ensure that justification is turned off, but *.SA* should be used to restore justification, rather than the *.ad* request so you can specify justification or no justification for the remainder of the text by inserting ".SA 0" or ".SA 1" once at the beginning of the document.

SCCS Release Identification

The *RE* string contains the SCCS release and the MM text formatting macro package current version level. For example:

```
This version \*(RE of the macros.
```

produces

This version 10.129 of the macros.

This information is useful in analyzing suspected bugs in MM. The easiest way to have the release identification number appear in the output is to specify `-rD1` on the command line. This causes the *RE* string to be output as part of the page header.

Two-Column Output

```
.2C
text and formatting requests (except another .2C)
.1C
```

The MM text formatting macro package can format two columns on a page. The *.2C* macro begins 2-column processing which continues until a *.1C* macro (1-column processing) is encountered. In 2-column processing, each physical page is treated as containing 2-columnar "pages" of equal (but smaller) "page" width. Page headers and footers are not affected by 2-column processing. The *.2C* macro does not balance 2-column output.

You can have full-page-width footnotes and displays when in 2-column mode, although the default is for footnotes and displays to be narrow in 2-column mode and wide in 1-column mode. Footnote and display width is controlled by the *.WC* (width control) macro, which takes the following arguments:

<arg>	Meaning
N	Default
WF	Wide footnotes (even in 2-column mode).
-WF	DEFAULT: Turn off WF. Footnotes follow column mode; wide in 1-column mode (1C), narrow in 2-column mode (2C), unless FF is set.
FF	First footnote. All footnotes have same width as first footnote encountered for that page
-FF	DEFAULT: Turn off FF. Footnote style follows settings of WF or -WF.
WD	Wide displays (even in 2-column mode).
-WD	DEFAULT: Displays follow the column mode in effect when display is encountered.
FB	DEFAULT: Floating displays cause a break when output on the current page.
-FB	Floating displays on current page do not cause a break.

Note

The `.WC WD EF` command produces wide displays and footnotes on the current page while `.WC N` reinstates default actions. If conflicting `.WC` settings are given, the last one encountered is used. `.WC WF -WF` is treated as `.WC -WF`.

Column Headings for Two-Column Output

Note

This section is intended only for users accustomed to writing formatter macros.

When handling 2-column output formats, it is sometimes necessary to have headers over each column, as well as headers over the entire page. This is accomplished by redefining the *.TP* macro to provide header lines for both the entire page and each of the two columns. For example:

```
.de TP
.sp 2
.tl 'Page \\nP'OVERALL"
.tl "TITLE"
.sp
.nf
.ta 16C 31R 34 50C 65R
left⇒center⇒right⇒left⇒center⇒right
⇒first column⇒⇒⇒second column
.fi
.sp 2
...
```

where ⇒ stands for the tab character.

This example produces two lines of page header text plus lines of headers over each column. Tab stops are for the 65-en overall line length.

Vertical Spacing

```
.SP [<lines>]
```

Several methods can be used to obtain vertical spacing, each with different effects. The *.sp* request spaces the number of <lines> specified unless the no-space (*.ns*) mode is on, in which case the *.sp* request is ignored. No-space mode is set at the end of a page header to eliminate spacing by a *.sp* or *.bp* request that happens to occur at the top of a page. No-space can be disabled by the *.rs* (restore spacing) request.

.SP is used to avoid the accumulation of vertical space by successive macro calls. Several successive *.SP* calls do not produce the sum of the arguments but only the maximum argument. For example, the following produces only three blank lines:

```
.SP 2
.SP 3
.SP
```

Many MM macros use *.SP* for spacing. For example, `.LE 1` immediately followed by `.P` produces only a single blank line (one-half of a vertical space) between the end of the list and the following paragraph. An omitted argument defaults to one blank line (one vertical space). Negative arguments are not permitted. The argument must be unscaled but fractional values are acceptable.

.SP (as well as *.sp*) is also inhibited by the *.ns* request.

Skipping Pages

```
.SK [<pages>]
```

The *.SK* macro skips pages but retains the usual header and footer processing. If the <pages> argument is omitted, null, or 0, *.SK* skips to the top of the next page unless it is currently at the top of a page (in which case it does nothing). `.SK` <n> skips <n> pages. For example, `.SK` always positions text that follows it at the top of a new page (skip to next page), while `.SK 1` always leaves one page blank except for the header and/or footer on the blank page.

Forcing an Odd Page

```
.OP
```

The *.OP* macro is used to ensure that formatted output text following the macro begins at the top of an odd-numbered page. If currently at the top of an odd-numbered page, text output begins on that page (no motion takes place). If currently on an even page, text resumes printing at the top of the next page. If currently on an odd page (but not at the top of the page), one blank page is produced, and printing resumes on the next odd-numbered page after that.

Setting Point Size and Vertical Spacing

```
.S [<point size>] [<vertical spacing>]
```

The default *troff* point size (obtained from the MM register *S* is 10 points, and the vertical spacing is 12 points (six lines per inch). Prevailing point size and vertical spacing can be changed by invoking the *.S* macro.

The mnemonics **D** (default value), **C** (current value), and **P** (previous value) can be used for both arguments.

- If an argument is negative, current value is decremented by the specified amount.
- If an argument is positive, current value is incremented by the specified amount.
- If an argument is unsigned, it is used as the new value.
- If there are no arguments, *.S* defaults to **P**.
- If the first argument is specified but the second is not, the (default) **D** is used for the vertical spacing.

Default value for vertical spacing is always two points greater than the current point size. Footnotes are two points smaller than the body with an additional 3-point space between footnotes. A null (" ") value for either argument defaults to C (current value). Thus, if n is a numeric value:

```
.S          =.S P P
.S""n       =.S C n
.Sn""       =.S n C
.Sn         =.S n D
.S""        =.S C D
.S"" ""     =.S C C
.S n n      =.S n n
```

If the first argument is greater than 99, the default point size (10 points) is restored. If the second argument is greater tha 99, the default vertical spacing (current point size plus two points) is used. For example:

```
.S 100      =.S 10 12
.S 14 111   =.S 14 16
```

You can use the *.SM* macro to reduce the size of a string by one point:

```
.SM <string1> [<string2>] [<string3>]
```

If <string3> is omitted, <string1> is made smaller and is concatenated with the <string2> if <string2> is specified. If <string1>, l<string2>, and <string3> are present (even if any are null), <string2> is made smaller and all three strings are concatenated. For example:

Input	Output
.SM X	X
.SM X Y	XY
.SM Y X Y	YXY
.SM YXYX	YXYX
.SM YXYX)	YXYX)
.SM (YXYX)	(YXYX)
.SM Y XYX ""	YXYX

Producing Accents

The following strings can be used to produce accents for letters:

	Input	Output
Grave accent	`c\*'`	è
Acute accent	`e\*'`	é
Circumflex	`o\*^`	ô
Tilde	`n\*-`	ñ
Cedilla	`c\*,`	ç
Lowercase umlaut	`u\*:`	ü
Uppercase umlaut	`U\*;`	Ü

Inserting Text Interactively

`.RD` [<prompt>] [<diversion>] [<string>]

The *.RD* (read insertion) macro stops standard document output and reads text from the standard input until two consecutive newline characters are found. When the two newline characters are encountered, normal output is resumed.

- <Prompt> is printed at the terminal. If not given, *.RD* signals the user with a BEL on terminal output.
- <Diversion> saves all text typed in after <prompt> in a macro whose name is that of the <diversion>.
- <String> saves the first line following the prompt in the named string for future reference.

.RD follows the formatting conventions currently in effect (the following example assumes that *.RD* is invoked in no-fill mode [*.nf*]). The following command:

```
.RD Name aA bB
```

combined with the following text from the standard input device:

```
Name:  J. Jones
16 Elm Rd.,
Piscataway
```

produces the following in the *.aA* macro:

```
J. Jones
16 Elm Rd.,
Piscataway
```

The string bB (`\*B`) contains "J. Jones".

A newline character from the input device followed by an EOF (user-specifiable CONTROL d) resumes normal output.

Errors and Debugging

Error Terminations

When a macro detects an error, the following sequence is followed:

1. A BREAK occurs.
2. The formatter output buffer (which may contain some text) is printed to avoid confusion regarding location of the error.
3. A short message is printed giving the name of the macro that detected the error, type of error, and approximate line number in the current input file of the last processed input line (error messages are explained in Table 4).
4. Processing terminates unless regiser *D* has a positive value, in which case processing continues even though the output is guaranteed to be incorrect from that point on.

The error message is printed by outputting the message directly to the user terminal. If an output filter, such as 300(1), 450(1), or hp(1) is being used to post-process the *nroff* formatter output, the message may be garbled as a result of being intermixed with text held in that filter's output buffer.

Note

If any of *cw*(1), *eqn*(1)/*neqn*, and *tbl*(1) programs are being used and if the `-o` list option of the formatter causes the last page of the document to not be printed, a harmless "broken pipe" message may result.

Disappearing Output

Output disappearances are usually the result of an unclosed diversion (such as a missing .DE or .FE macro call). Fortunately, macros that use diversions are careful about it, and these macros check to make sure that illegal nestings do not occur. If any error message is issued concerning a missing .DE or .FE, the appropriate action is to search backwards from the termination point looking for the corresponding associated .DF, .DS, or .FS (since these macros are used in pairs).

The following command:

```
grep -n '^\.[EDFR]EFNQ]' <files> . . .
```

prints all the .DF, .DS, .DE, .EQ, .EN, .FS, .FE, .RS, .RF, .TS, and .TE macros found in *files* ..., each preceded by its file name and the line number in that file. This listing can be used to check for illegal nesting and/or omission of these macros.

Extending and Modifying MM Macros

Naming Conventions

In this part, the following conventions are used to describe names:

n:	Digit
a:	Lowercase letter
A:	Uppercase letter
x:	Any alphanumeric character (:a, n, A, or 1, ie., letter or digit)
s:	any nonalphanumeric character (special character)

All other characters are literals (i.e., characters stand for themselves).

Request, macro, and string names are kept in a single internal table by the formatters. Therefore, there must be no duplication among such names. Number register names are kept in a separate table.

Names Used by Formatters

requests:	aa (most common) an (currently only one: c2)
registers:	aa (normal) .x (normal) .s (currently only one: .$) a. (currently only one: c.) % (page number)

Names Used by MM

macro and strings:	A, AA, Aa (accessible to users; e.g., macros P and HU, strings F, BU, and Lt) nA (accessible to users; currently only two: 1C and 2C) aA (accessible to users; currently only one: nP) s (accessible to users; currently only the seven accents))x, }x,]x, >x, ?x (internal)
register	An, Aa (accessible to users; e.g., H1, Fg) A (accessible to users; meant to be set on the command line; e.g., C) :x, ;x, #x, ?x, !x (internal)

Names Used by CW, EQN/NEQN, and TBL Programs

The *cw*(1) program is the constant-width font preprocessor for *troff*. It uses the following five macro names:

`.CD`, `.CN`, `.CP`, `.CW`, and `.PC`.

This preprocessor also uses the number register names `cE` and `cW`. Mathematical equation preprocessors, *eqn*(1) and *neqn*, use registers and string names of the form *nn*. The table preprocessor, *tbl*(1), uses `T&`, `T#`, and `TW`, and names of the form:

a- a+ al nn na ^a #a #s

Names Defined by User

Names that consist either of a single lowercase letter or a lowercase letter followed by a character other than a lowercase letter (names *.c2* and *.nP* are already used) should be used to avoid duplication with already used names. Here is a possible naming convention.

macros:	aA (e.g., *bG*, *kW*)
strings:	as (e.g., *c)*, *f]*, *p}*)
registers:	a (e.g., *f,t*)

Sample Extensions

Appendix Headings

Here is a method for generating and numbering appendix headings:

```
.nr Hu 1
.nr a 0
.de aH
.nr a+1
.nr P 0
.PH"'''Appendix\\na-\\\\\\\\nP'"
.SK
.HU "\\$1"
...
```

After the above initialization and definition, each call of the form `.aH "title"` begins a new page (with the page header changed to "Appendix a-n" and generates an unnumbered heading "*title*", which can be saved for the table of contents, if desired. To center appendix titles, set register *Hc* to 1.

Hanging Indent With Tabs

The following example illustrates the use of the hanging indent feature of variable-item lists. A user-defined macro is defined to accept four arguments that make up the mark. In the output, each argument is to be separated from the previous one by a tab (tab settings are set by *.ta* request and `\t` is interpreted as a tab key by formatters). Since the first argument can begin with a period or apostrophe, the "\&" is prepended to it so that the formatter will not interpret such a line as a formatter request or macro call.

Note

Formatters interpret the 2-character sequence "\&" as a "zero-width" space. This provides a method for removing the the special meaning of leading periods and apostrophes without affecting output text. Formatters translate `\t` into a tab, and `\c` is used to concatenate input text that follows the macro call to the line built by the macro.

Here is the macro and an example of its use:

```
.de aX
.LI
\&\\$1\t\\$2\t\\$3\t\\$4\t\c
...
  .
  .
  .
.ta 8 14 20 24
.VL
.aX .nh off\-no
No hyphenation.
Automatic hyphenation is turned off.
Words containing hyphens
(e.g., mother-in-law) may still be split across lines.
.aX .hy on \-no
Hyphenate.
Automatic hyphenation is turned on.
.aX .hc\<sp>c none none no
Hyphenation indicator character is set to "c" or
removed.
During text processing, the indicator is suppressed
and will not appear in the output
Prepending the indicator to a word has the effect
of preventing hyphenation of that word.
.LE
```

where <sp> stands for a space.

The resulting output is:

.nh	off	—	no	No hyphenation. Automatic hyphenation is turned off. Words containing hyphens (e.g., mother-in-law) may still be split across line.
.hy	on	-	no	Hyphenate. Automatic hyphenation is turned on.
hc c	none	none	no	Hyphenation indicator character is set to "c" or removed. During text processing, the indicator is suppressed and will not appear in the output. Prepending the indicator to a word has the effect of preventing hyphenation of that word.

Summary

The following are qualities of MM that have been emphasized in its design in approximate order of importance:

- *Robustness in the face of error* – A user need not be an *nroff/troff* expert to use MM macros. When the input is incorrect, either the macros attempt to make a reasonable interpretation of the error or an error message describing the error is produced. An effort has been made to minimize the possibility that a user would get cryptic system messages or strange output as a result of simple errors.
- *Ease of use for simple documents* – It is not necessary to write complex command sequences to produce documents. Reasonable macro argument default values are provided where possible.
- *Parameter based* – There are many different preferences in the area of document styling. Many parameters are provided so that users can adapt input text files to produce output documents to meet their respective needs with a wide range of styles.
- *Extension by moderately expert users* – A strong effort has been expended to use mnemonic naming conventions and consistent techniques in construction of macros. Naming conventions are given so that a user can add new macros or redefine existing ones if necessary.
- *Device independence* – A common use of MM is to produce documents on hard copy via printing terminal using the *nroff* formatter. Macros can be conveniently used with both 10- and 12-pitch terminals. In addition, output can be displayed on an appropriate CRT trminal. Macros have been constructed to provide compatibilty with the *troff*(1) formatter so that output can be produced on phototypesetters and printing/CRT terminals.
- *Minimization of input* – The design of macros attempts to minimize repetitive typing. For example, to produce a blank line after all first- or second-level headings, only one parameter is required, and it is set at the beginning of the document; eliminating the need for a blank line after each heading affected.
- *Decoupling input format from output style* – There is but one way to prepare the input text although the user can obtain a number of output styles by setting a few global flags. For example, the *.H* macro is used for all numbered headings, yet the actual output style of these headings can be varied from document to document or within a single document.

Footnote Examples

Input Text File:

```
.P
.FD 10
This example illustrates several footnote styles
for both labelled and automatically numbered footnotes.
With the footnote style set to the NROFF default,
process the first footnote\*F
.FS
This is the first footnote text example (.FD 10).
This is the default style for NROFF.
The right margin is not justified.
Hyphenation is not permitted.
Text is indented, and the automatically generated label is
right justified in the text-indent space.
.FE
and follow it by a second footnote.*****
.FS*****
This is the second footnote text example (.FD 10).
This is also the default NROFF style but with a long
footnote label (*****) provided by the user.
.FE
.FD 1
Footnote style is changed by using the .FD' macro to
specify hyphenation, right margin justification,
indentation, and left justification of the label.
This produces the third footnote, \*F
.FS
This is the third footnote example (.FD 1).
The right margin is justified, the footnote text is indented,
and the label is left justified in the text-indent space.
Although not necessarily illustrated by this example,
hyphenation is permitted.
.FE
and then the fourth footnote.\(dg
.FS
This is the fourth footnote example (.FD 1).
The style is the same as the third footnote.
.FE
.FD 6
Footnote style is set again via the .FD macro for no hyphenation,
no right margin justification,
no indentation, and with the label left justified.
This produces the fifth footnote. \*F
.FS
This is the fifth footnote example (.FD 6).
The right margin is not justified, hyphenation is not pemitted,
footnote text is not indented,
and the label is placed at the beginning of the first line.
.FE
```

NROFF Printed Output:

```
This example illustrates several footnote styles for both
labelled and automatically numbered footnotes.  With the
footnote style set to the NROFF default, process the first
footnote and follow it by a second footnote.***** Footnote
style is changed by using the .FD macro to specify
hyphenation, right margin justification, indentation, and
left justification of the label.  This produces the third
footnote, and then the fourth footnote.  Footnote style is
set agin via the .FD macro for no hyphenation, no right
margin justification, no indentation, and with the label
left justified.  This produces the fifth footnote.
```

```
1.  This is the first footnote text example (.FD 10).  This
    is the default style for NROFF.  The right margin is not
    justified.  Hyphenation is nt permitted.  Text is
    indented, and the automatically generated label is right
    justified in the text-indent space.

***** This is the second footnote text example (.FD 10).
    This is also the default NROFF style but with a long
    footnote label (*****) provided by the user.

2.  This is the third footnote example (.FD 1).  The right
    margin is justified, the footnote text is indented, and
    the label is left justified in the text-indent space.
    Although not necessariy illustrated by this example,
    hyphenation is permitted.

+   This is the fourth footnote example (.FD 6).  The right
    margin is not justified, hyphenation is not permitted,
    footnote text is not indented, and the label is placed at
    the beginning of the first line.

3. This is the fifth footnote example (.FD 6).  The right
margin is not justified, hyphenation is not permitted,
footnote text is not indented, and the label is placed at
the beginning of the first line.
```

Tables

This section contains tables of reference information including macro names, string and numeric registers, and error messages. Numbers inside braces ({ }) refer to the page number where the topic is explained in greater detail.

Table 1: Mm Macro Names Sumary

Macro	Description {Page} Command Syntax
1C	1-column processing {59} .1C
2C	2-column processing {59} .2C
AE	Abstract end {35} .AE
AF	Alternate format of "Subject/Date/From" block {37} .AF [company-name]
AL	Automatically incremented list start {23} .AL [type] [text-indent] [1]
AS	Abstract start {35} .AS [arg] [indent]
AT	Author's title {34} .AT [title]...
AU	Author information {34} .AU name [initials] [loc] [dept] [ext] [room] [arg] [arg] [arg]
AV	Approval signature {40} .AV [name]
B	Bold {58} .B [bold-arg] [previous-font-arg] [bold] [prev] [bold] [prev]
BE	Bottom block end {52} .BE
BI	Bold/Italic {58} .BI [bold-arg] [italic-arg] [bold] [italic] [bold] [italic]
BL	Bullet list start {24} .BL [test-indent] [1]
BR	Bold/Roman {58} .BR [bold-arg] [Roman-arg] [bold] [Roman] [bold] [Roman]
BS	Bottom block start {52} .BS

Table 1: Mm Macro Names Summary (continued)

Macro	Description {Page} Command Syntax
CS	Cover sheet {56} .CS [pages] [other] [total] [figs] [tbls] [refs]
DE	Display end {41} .DE
DF	Display floating start {42} .DF [format] [fill] [right-indent]
DL	Dash list start {25} .DL [text-indent] [1]
DS	Display static start {41} .DS [format] [fill] [right-indent]
EC	Equation caption {46} .EC [title] [override] [flag]
EF	Even-page footer {50} .EF [arg]
EH	Even-page header {50} .EH [arg]
EN	End equation display {45} .EN
EQ	Equation display start {45} .EQ [label]
EX	Exhibit caption {46} .EX [title] [override] [flag]
FC	Formal closing {38} .FC [closing]
FD	Footnote default format {48} .FD [arg] [1]
FE	Footnote end {47} .FE
FG	Figure title {46} .FG [title] [override] [flag]
FS	Footnote start {47} .FS [label]
H	Heading – numbered {15} .H level [heading-text] [heading-suffix]
HC	Hyphenation character {10} .HC [hyphenation-indicator]
HM	Heading mark style {18} (Aragic or Roman numerals, or letters) .HM [arg1]...[arg7]
HU	Heading – unnumbered {19} .HU heading-text

Table 1: Mm Macro Names Summary (continued)

Macro	Description {Page} Command Syntax
HX[1]	Heading user-exit X (before printing heading) {20} .HX dlevel rlevel heading-text
HY[1]	Heading user-exit Y (before printing heading) {20} .HY dlevel rlevel heading-text
HZ[1]	Heading user-exit Z (after printing heading) {20} .HZ dlevel rlevel heading-text
I	Italic (underline in the *nroff* formatter) {58} .I [italic-arg] [previous-font-arg] [italic] [prev] [italic] [prev]
IB	Italic/Bold {58} .IB [italic-arg] [bold-arg] [italic] [bold] [italic] [bold]
IR	Italic/Roman {58} .IR [italic-arg] [Roman-arg] [italic] [Roman] [italic] [Roman]
LB	List begin {29} .LB text-indent mark-indent pad type [mark] [LI-space] [LB-space]
LC	List-status clear {32} .LC [list-level]
LE	List end {28} .LE [1]
LI	List item {27} .LI [mark] [1]
ML	Marked list start {25} .ML mark [text-indent] [1]
MT	Memorandum type {36} .MT [type] [addressee] or .MT [4] [1]
ND	New date {37} .ND new-date
NE	Notation end {39} .NE
NS	Notation start {39} .NS [arg]
nP	Double-line indented paragraphs {15} .nP
OF	Odd-page footer {50} .OF [arg]
OH	Odd-page header {50} .OH [arg]
OK	Other keywords for the Technical Memorandum cover sheet {36} .OK [keyword]...

1 See note at end of table

Table 1: Macro Names Summary (continued)

Macro	Description {Page} Command Syntax
OP	Odd page {61} .OP
P	Paragraph {14} .P [type]
PF	Page footer {50} .PF [arg]
PH	Page header {50} .PH [arg]
PM	Proprietary marking {53} .PM [code]
PX[1]	Page-header user exit {52} .PX
R	Return to regular (Roman) font {58} .R
RB	Roman/Bold {58} .RB [Roman-arg] [bold-arg] [Roman] [bold] [Roman] [bold]
RD	Read insertion from terminal {63} .RD [prompt] [diversion] [string]
RF	Reference end {56} .RF
RI	Roman/Italic {58} .RI [Roman-arg] [italic-arg] [Roman] [italic] [Roman] [italic]
RL	Reference list start {25} .RL [text-indent] [1]
RP	Produce reference page {57} .RP [arg] [arg]
RS	Reference start {56} .RS [string-name]
S	Set *troff* formatter point size and vertical spacing {62} .S [size] [spacing]
SA	Set adjustment (right-margin justification) default {59} .SA [arg]
SG	Signature line {38} .SG [arg] [1]
SK	Skip pages {61} .SK [pages]

1 See note at end of table.

Table 1: Mm Macro Names Summary (continued)

Macro	Description {Page} Command Syntax
SM	Make a string one point-size smaller {62} .SM string1 [string2] [string3]
SP	Space vertically {61} .SP [lines]
TB	Table title {46} .TB [title] [override] [flag]
TC	Table of contents {54} .TC [slevel] [spacing] [tlevel] [tab] [head1] [head2] [head3] [head4] [head5]
TE	Table end {44} .TE
TH	Table header {44} .TH [N]
TL	Title of memorandum {33} .TL [charging-case] [filing-case]
TM	Technical Memorandum number(s) {35} .TM [number]...
TP[1]	Top-of-page macro {52} .TP
TS	Table start {44} .TS [H]
TX[1]	Table of contents user exit {55} .TX
TY[1]	Table of contents user exit {55} (suppresses "CONTENTS") .TY
VL	Variable-item list start {25} .VL text-indent [mark-indent] [1]
VM	Vertical margins {53} .VM [top] [bottom]
WC	Footnote and Display Width control {60} .WC [format]

1 Macros referencing this footnote are not generally called (invoked) by users. They are "user exits" defined by the user and called by MM from within header, footer, or other macros.

Table 2: String Names Summary

String Name	Description {page}
BU	Bullet {11}
Ci	Table of contents indent list {54} Up to seven args (must be scaled) for heading levels 1 through 7
DT	Date {37} Current date (unless overridden) Month, Day, Year (e.g., July 16, 1986)
EM	Em dash string {12} Produces an em dash in the *troff* formatter and a double hyphen in *nroff*
F	Footnote numberer { } *NROFF*: `\u\\n+(:P\d` *TROFF*: `\v'-.4m'\s-3\\n+(:*P\s0\v'.4m'`
HF	Heading font list {17} Up to seven codes for heading levels 1 through 7 Defaults: 3 3 2 2 2 2 2 (levels 1 and 2 bold, 3 through 7 underlined in the *nroff* formatter and italic in *troff*)
HP	Heading point size list {18} Up to seven codes for heading levels 1 through 7
Le	Title for LIST OF EQUATIONS {46}
Lf	Title for LIST OF FIGURES {46}
Lt	Title for LIST OF TABLES {46}
Lx	Title for LIST OF EXHIBITS {46}
RE	SCCS Release and Level of MM {59} Release.Level (e.g., 10.129)
Rf	Reference numberer {56}
Rp	Title for references {57}
Tm	Trademark string {12} Places the letters "TM" one-half line above the text that follows. Seven accent strings are also available {63}

Note 1: If the released-paper style is used, then, in addition to the above strings, certain Bell Telephone Laboratories location codes are defined as strings; these location strings are not used after the .MT macro is called {36}. The following are recognized:

AK, AL, ALF, CB, CH, CP, DR, FJ, HL, HO, HOH, HP, IH, IN, INH, IW, MH, MV, PY, RD, RR, WB, WH, and WV.

Note 2: See page 3 for notes on setting and referencing strings.

Table 3: Numeric Register Names Summary

Register	Description {Page} Default, [Allowable-value range]
A[2]	Handles preprinted forms and Bell System logo {7} 0, [0:2]
Au	Inhibits printing author's location, department, room, and extension in "from" portion of memorandum {34}
C[2]	Copy type {7} Original, Draft, etc. 0 (Original), [0:4]
C1	Contents level {19} Level of headings saved for table of contents 2, [0:7]
Cp	Placement of list of figures, etc. {55} 1 (on separate pages), [0:1]
D[2]	Debug flag {7} 0,[0:1]
De	Display eject register for floating displays {43} 0, [0:1]
Df	Display format register for floating display {43} 5, [0:5]
Ds	Static display pre- and post-space {42} 1, [0:1]
E[2]	Controls font of the Subject/Date/From fields {7} 1 (*nroff*) 0 (*troff*), [0:1]
Ec	Equation counter, used by .EC macro {46} 0, [0:?], incremented by one for each .EC call.
Ej	Page-ejection flag for headings {16} 0 (no eject), [0:7]
Eq	Equation label placement {45} 0 (right-adjusted), [0:1]
Ex	Exhibit counter, used by .EX macro {46} 0, [0:?], incremented by one for each .EX call.
Fg	Figure counter, used by .FG macro {46} 0, [0:?], incremented by one for each .FG call.
Fs	Footnote space (i.e., spacing between footnotes) {49} 1, [0:?]
H1 through H7	Heading counters for levels 1 through 7 {18} 0, [0:?], incremented by .H of corresponding level or .HU if at level given by register Hu. H2 through H7 are reset to 0 by any heading at a lower-numbered level.

2 See notes at end of table.

Table 3: Numeric Register Names Summary

Register	Description {Page} Default, [Allowable-value range]
Hb	Heading break level (after .H and .HU) {16}
Hc	Heading centering level for .H and .HU {17} 0 (no centered headings), [0:7]
Hi	Heading temporary indent (after .H and .HU) {16} 1 (indent as paragraph), [0:2]
Hs	Heading space level (after .H and .HU) {16} 2 (space only after .H 1 and .H 2), [0:7]
Ht	Heading type {19} For .H: single or concatenated numbers 0 (concatenated numbers: 1.1.1, etc.), [0:1]
Hu	Heading level for unnumbered heading (.HU) {19} 2 (.HU at the same level as .H 2, [0:7]
Hy	Hyphenation control for body of document {10} 0 (atuomatic hyphenation off), [0:1]
L[2]	Length of page {7} 66, [20:?] (11i, [2i:?] in *troff* formatter) For *nroff*, these values are unscaled numbers representing lines or character positions; for *troff*, values must be scaled.
Le	List of equations {46} 0 (list not produced) [0:1]
Lf	List of figures {46} 1 (list not produced) [0:1]
Li	List indent {24} 6 (nroff) 5 (troff), [0:?]
Ls	List spacing between items by level {24} 6 (spacing between all levels) [0:6]
Lt	List of tables {46} 1 (list produced) [0:1]
LX	List of exhibits {46} 1 (list produced) [0:1]
N[2]	Numbering style {7} 0, [0:5]
Np	Numbering style for paragraphs {15} 0 (unnumbered) [0:1]

2 See notes at end of table.

Table 3: Numeric Register Names Summary

Register	Description {Page} Default, [Allowable-value range]
O[2]	Offset of page {8} .75i, [0:?] (0.5i, [0i:?] in *troff* formatter) For *nroff* formatter, these values are unscaled numbers representing lines or character positions; for *troff* formatter, these values must be scaled.
Oc	Table of contents page numbering style {55} 0 (lowercase Roman), [0:1]
Of	Figure caption style {46} 0 (period separator), [0:1]
P[2]	Page number manager by MM {8} 0,.[0:?]
Pi	Paragraph indent {14} 5 (*nroff*) 3 (*troff*), [0:?]
Ps	Paragraph spacing {15} 1 (one blank space between paragraphs), [0:?]
Pt	Paragraph type {14} 0 (paragraphs always left justified), [0:2]
Pv	"PRIVATE" header {53} 0 (not printed), [0:2]
Rf	Reference counter, used by .RS macro {56} 0, [0:?], incremented by one for each .RS call.
S*[2]	The *troff* formatter default point size {8} 10, [6:36]
Si	Standard indent for displays {42} 5 (*nroff*) 3 (*troff*), [0:?]
T*[2]	Type of *nroff* output device {8} 0, [0:2]
Tb	Table counter, used by .TB macro {46} 0, [0:?], incremented by one for each .TB call.
U*[2]	Underlining style (*nroff*) for .H and .HU {8} 0 (continuous underline when possible), [0:1]
W*[2]	Width of page (line and title length) {8} 6i, [10:1365] (6i, [2i:7.54i] in the *troff* formatter)

* An asterisk attached to a register name indicates that this register can be set only from the command line or before the MM macro definitions are read by the formatter {6, 7, 8}.

2 Page 3 has notes on setting and referencing registers. Any register having a single-character name can be set from the command line.

Table 4: Error Messages

An MM error message has a standard part followed by a variable part. The standard part has the form:

```
ERROR: (filename) input line n:
```

Variable parts consist of a desciptive message usually beginning with a macro name. They are listed here in alphabetical order by macro name, each with a more complete explanation.

Mm Error Messages

Error Message	Description
Check TL, AU, AS, AE, MT sequence	The correct order of macros at the start of a memorandum is shown in {..}. Something has disturbed this order.
Check TL, AU, AS, AE, NS, NE, MT sequence	The correct order of macros at the start of a memorandum is shown in {..}. Something has disturbed this order. Occurs if the .AS 2 {..} macro was used.
CS:cover sheet too long	Text of the cover sheet is too long to fit on one page. The abstract should be reduced or the indent of the abstract should be decreased { }.
DE:no DS or DF active	A .DE macro has been encountered, but there has not been a previous .DS or .DF macro to match it.
DF:illegal inside TL or AS	Displays are not allowed in the title or abstract.
DF:missing DE	A .DF macro occurs within a display, i.e., a .DE macro has been omitted or mistyped.
DF:missing FE	A display starts inside a footnote. The likely cause is the omission (or misspelling) of a .FE macro to end a previous footnote.
DF:too many displays	More than 26 floating displays are active at once, i.e., have been accumulated but not yet output.
DS:illegal inside TL or AS	Displays are not allowed in the title or abstract.
DS:missing DE	A .DS macro occurs within a display, i.e., a .DE has been been omitted or mistyped.
DS:missing FE	A display starts inside a footnote. The likely cause is the omission (or misspelling) of a .FE to end a previous footnote.
FE:no FS active	A .FE macro has been encountered with no previous .FS to match it.
FS:missing DE	A footnote starts inside a display, i.e., a .DS or .DF occurs without a matching .DE.
FS:missing FE	A previous .FS macro was not matched by a closing .FE, i.e., an attempt is being made to begin a footnote inside another one.
H:bad arg: <value>	The first argument to the .H macro must be a single digit from one to seven, but <value> has been supplied instead.
H:missing arg	The .H macro needs at least one argument.
H:missing DE	A heading macro (.H or .HU) occurs inside a display.
H:missing FE	A heading macro (.H or .JU) occurs inside a footnote.

Table 4: Error Messages (continued)

***Mm* Error Messages**

Error Message	Description
HU:missing arg	The .JU macro needs one argument.
LB:missing arg(s)	The .LB macro requires at least four arguments.
LB:too many nested lists	Another list was started when there were already six active lists.
LE:mismatched	The .LE macro has occurred without a previous .LB or other list-initialization macro {28}. Although this is not a fatal error, the message is issued because there almost certainly exists some problem in the preceding text.
LI:no lists active	The .LI macro occurred without a preceding list-initialization macro. The latter has probably been omitted or has been separated from the .LI by an intervening .H or .HU.
ML:missing arg	The .ML macro requires at least one argument.
ND:missing arg	The .ND macro requires one argument.
RF:no RS active	The .RF macro has been encountered with no previous .RS to match it.
RP:missing RF	A previous .RS macro was not matched by closing .RF.
RS:missing RF	A previous .RS macro was not matched by a closing .RF.
S:bad arg: <value>	The incorrect argument <value> has been given for the .S macro {..}.
SA:bad arg:<value>	The argument to the .SA macro (if any) must be either 0 or 1. the incorrect argument is shown as <value>.
SG:missing DE	The .SG macro occurred inside a display.
SG:missing FE	The .SG macro occurred inside a footnote.
SG:no authors	The .SG macro occurred without any previous .AU macro(s).
VL:missing arg	The .VL macro requires at least one argument.
WC:unknown option	An incorrect argument has been given to the .WC macro {60}.

Formatter Error Messages

Most messages issued by the formatter are self-explanatory. Those error messages over which the user has some control are listed below. Any other error messages should be reported to the local system support group.

Error Message	Description
Cannot do ev	Caused by: a. setting a page width that is negative or extremely short b. setting a page length that is negative or extremely short c. reprocessing a macro package (e.g., performing a .so request on a macro package that was already requested on the command line) d. requesting the *troff* formatter (an option on a document that is longer than ten pages).

Table 4: Error Messages (continued)

Error Message	Description
Cannot execute <filename>	Given by the .! request if the <filename> is not found.
Cannot open <filename>	Indicates one of the file in the list of files to be processed cannot be opened.
Exception word list full	Indicates too many words have been specified in the hyphenation exception list (vis .hw requests)
Line overflow	Indicates output line being generated was too long for the formatter line buffer capacity. The excess was discarded. Likely causes for this message are very long lines or words generate through the misuse of \c of the .cu request or very long equations produced by *eqn(1)/neqn.*
Non existent font type	Indicates a request has been made to mount an unknown font.
Nonexistent macro file	Indicates the requested macro package does not exist.
Nonexistent terminal type	Indicates the terminal options refer to an unknown terminal type.
Out of temp file space	Indicates additional temporary space for macro definitions, diversions, etc. cannot be allocated. This message often occurs because of unclosed diversions (missing .FE or .DE), unclosed macro definitions (e.g., missing ".."), or a huge table of contents.
Too many page numbers	Indicates the list of pages specified to the -o formatter option is too long.
Too many number registers	Indicates the pool of number register names is full. Unneeded registers can be deleted by using the .rr request.
Too many string/macro names	Indicates the pool of string and macro names is full. Unneeded strings and macros can be deleted using the .rm request.
Word overflow	Indicates a word being generated exceeds the formatter word buffer capacity. Excess characters were discarded. Likely causes for this message are very long lines, words generated through the misuse of \c of the .cu request, or very long equations produced by *eqn (1)/neqn.*

Table of Contents

Tbl
A Program to Format Tables

Tbl is an HP-UX document formatting command that is used as a preprocessor for *troff* or *nroff* formatters, making even fairly complex tables easy to specify and enter.

Tables are made up of columns which can be independently centered, right-adjusted, left-adjusted, or aligned by decimal point location. Headings can be placed over single columns or groups of columns. A table entry can contain equations, and can accommodate multiple rows of text. Horizontal or vertical lines can be drawn as desired in the table, and any table or element can be enclosed in a box.

Tbl takes input text containing *tbl* commands and data, processes it, and produces output that is compatible with *troff* and *nroff* formatters. If the source contains commands and data for other formatters, *tbl* processes only those parts of the job that can be successfully handled, leaving the remainder for other commands and programs supported on HP-UX.

Usage

To use *tbl* for constructing a simple table, execute the following HP-UX command:

`tbl` *filename* `|troff` **RETURN** (or (ENTER))

For more complex situations involving multiple files and/or *ms* or *mm* macro requests as well as tables, the normal command is:

`tbl` *file1 file2* `...troff -ms` **RETURN** (or (ENTER))

The usual options can be used on *troff*. *Nroff* is similar to *troff*, but many display and printing devices driven by *nroff* cannot reproduce boxed tables. If a file name is "-", the standard input device is used at that point.

For line printers that do not have adequate driving tables or post-filters, the special $-TX$ command-line option to *tbl* produces output having no fractional line motions. The only other command-line options recognized by *tbl* are $-ms$ and $-mm$. They are converted into commands to fetch the corresponding macro files. While it is usually more convenient to place these arguments on the *troff* formatter part of the command line, they are also accepted by *tbl*.

Whenever *eqn* and *tbl* are used together on the same file, *tbl* should be used first. If there are no equations within the table(s), either sequence works, but, since *eqn* usually produces a larger expansion of the input, it is usually faster to execute *tbl* first. On the other hand, if the table contains equations, *tbl* **must** be used first to prevent scrambling the output. Avoid using equations in *n*-style columns because *tbl* attempts to split numerical format items into two parts. The *delim*<xy> table option prevents splitting numerical columns between delimiters. For example, if the *eqn* delimiters are `$$` (set by `delim $$`), a numerical column such as 1245 $ ± 16$ is divided after 1245, not after 16.

Tbl accepts up to 35 columns, but the actual number that can be processed may be less, depending on the availability of *troff* formatter number registers. Number register names used by *tbl* **must** be avoided in tables. These include 2-digit numbers from 31 thru 99 and strings of the form 4x, 5x, *x, x+, x|, ^x, and x-, where x is any lowercase letter. The names **, *-, and *^ are also used in certain circumstances. To conserve register names, the *n* and *a* key-letters (key-letters are discussed in "Format Section" later in this tutorial) share a register; hence, the restriction that they cannot be used in the same column.

As an aid in writing layout macros, *tbl* defines a number register TW which is the table width. The TW number register is defined before the *.TE* macro is invoked, and can be used in the *.TE* expansion.

Another important macro *T#* defines and produces the bottom lines and side lines of a boxed table when it is invoked at the end of a table or page. Including this macro in the page footer enables you to create multi-page tables with headers repeated on each successive page in the table by using the H argument with the *.TS* macro. If the `table start` macro is written as follows:

```
.TS H
```

a line of the form

```
.TH
```

must be given in the table after any table heading (or at the start if no heading exists). Material preceding the .TH is placed at the top of each page of the table. The remainder of the table is placed on one or more pages, as needed. This feature is provided by the *ms* and *mm* macros, not by *tbl.*

Input Commands

The input to *tbl* is text for a document, where tables are preceded by *.TS* (table start command) and followed by *.TE* (table end command). *Tbl* processes the tables, generating *troff* formatting commands, and leaves the remainder of the text unchanged. The *.TS* and *.TE* lines are copied, too, so that *troff* page layout macros (such as the memo formatting macros) can use these lines to delimit and place tables as they see fit. In particular, any arguments on the *.TS* or *.TE* lines are copied but otherwise ignored, and can be used by document-layout macro commands.

The format of the input is as follows:

```
text
.TS
table
.TE
text
.TS
table
.TE
text
. . .
```

where the format of each *table* is:

```
.TS
options;
format.
data
.TE
```

Each table structure is independent, and contains:

- Global options,
- A format section describing individual columns and rows in the table,
- Data to be printed in the table

The format section and data are required, but the options list can be omitted.

Global Options

A single line of options affecting the entire table can be placed immediately following the `.TS` line. It must contain a list of option names separated by spaces, tabs, or commas, and must be terminated by a semicolon. Recognized options include:

center	Center the table (default is left-adjust).
expand	Make the table as wide as the current line length.
box	Enclose the table in a box.
allbox	Enclose each table item in a box.
doublebox	Enclose the table in two boxes.
tab <x>	Use <x> instead of tab to separate data items where <x> is any character.
linesize <n>	Set lines or rules (e.g., from *box*) in <n> point type.
delim <xy>	Recognize <x> and <y> as the *eqn* delimiters (<x> and <y> are any character).

Tbl tries to keep boxed tables on one page by issuing appropriate *.ne* (need) requests. These requests are calculated from the number of lines in the tables. If there are spacing commands embedded in the input, the *.ne* requests may be inaccurate, in which case, normal *troff* procedures, such as keep-release macros should be used. If you need a multi-page boxed table, use *.TS H* and *.TH* macros which are designed for this purpose.

Format Section

The format section of the table specifies column layout. Each line in the format section corresponds to one line of table data (except that the last format line corresponds to all the following data lines up to the next *.T&* command line. Each line contains a **key-letter** for each column of the table. **Key-letter**s for each column can be separated by spaces or tabs for better readability. Allowable key letters include:

L or *l*	Indicates a left-adjusted column entry.
R or *r*	Indicates a right-adjusted column entry.
C or *c*	Indicates a centered column entry.
N or *n*	Indicates a numerical column entry to be aligned with other numerical entries so that the units digits of numbers line up.
A or *a*	Indicates an alphabetic subcolumn where all corresponding entries are aligned on the left, and positioned so that the widest is centered within the column (see example later in this tutorial).

S or *s* Indicates a spanned heading; i.e., the previous-column continues across this column (not allowed for first column).

^ Indicates a vertically spanned heading; i.e., the previous-row entry continues down through this row (not allowed for first row).

When numerical column alignment (*n*) is specified, a location for the decimal point is sought. The right-most dot (.) adjacent to a digit is used as a decimal point unless there is no dot adjoining a digit, in which case the rightmost digit is used as a units digit. If no alignment is indicated, the item is centered in the column. However, the special non-printing character string `\&` can be used to unconditionally override dots and digits or align alphabetic data (strings). When included in a string, the `\&` is positioned such that the `&` is placed in the location that would otherwise be occupied by a dot, but the `\&` is suppressed before sending the final output.

In the following example, input items on the left are aligned on the right as they would be in a numerical table column.

```
13                          13
4.2                          4.2
26.4.12                    26.4.12
abcdefg                    abcdefg
abcdefg\&                abcdefg
43\&3.22                     433.22
749.12                       749.12
```

Note

When numerical data is found in the same column with wider *L*- or *r*-type table entries, the widest <number> is centered relative to the wider *L* or *r* items (*L* is used instead of *l* for readability; they have the same meaning as key-letters). Alignment within the numerical items is preserved. This is similar to the behavior of *a* type data, as previously explained. However, alphabetic subcolumns (requested by the *a* key-letter) are always slightly indented relative to *L* items (if necessary, the column width is increased to force this). This is not true for *n*-type entries.

Do not use *n* and *a* items in the same column.

For readability, separate the key-letters describing each column. The end of the format section is indicated by a period. The layout of the key-letters in the format section resembles the layout of the actual data in the table. Thus, a simple 3-column format might appear as:

```
css
lnn.
```

The first line specifies a heading centered across all three columns. Each remaining line contains a left-adjusted item in the first column followed by two columns of numerical data. Here is an example table using such a format:

```
c s s
l n n.
Overall title
Item a
34.22
9.1
Item b
12.65
.02
Items: c,d,e
23
5.8
Total
69.87
14.92
```

Here are some additional features of the key-letter system:

Horizontal lines

A key-letter can be replaced by underscore (_) to indicate a horizontal line in place of the corresponding column entry, or by equals (=) to indicate a double horizontal line. If an adjacent column contains a horizontal line, or if there are vertical lines adjoining the current column, the horizontal line is extended to meet both nearby lines. If any data is provided for this column of the current table, it is ignored and an error message is printed.

Vertical lines

A vertical bar can be placed between column key-letters, producing a vertical line between the corresponding columns of the table. A vertical bar to the left of the first key-letter of the current line and/or to the right of the last key-letter of the current line produces a line at the corresponding edge of the table. If two vertical bars appear between key-letters, a double vertical line is drawn.

Space between columns

A number can follow the key-letter, specifying the amount of separation between the current column and the next column. The number normally specifies the separation in *en* spaces where one *en* is about the width of the letter "n" (more precisely, the width of an en space is equal to half the number of points (1 point = 1/72 inch) in the current type size).

If the **expand** option is used, these numbers are multiplied by a constant such that the table is as wide as the current line length. The default column separation number is 3. If the separation is changed, worst case (largest space requested) governs.

Vertical spanning

Vertically spanned items extending over several rows of the table are centered in their vertical range. If a key-letter is followed by `t` or `T`, any corresponding vertically spanned item will begin at the top line of its range.

Font changes

A key-letter followed by a string containing a font name or number preceded by the letter *f* or *F* indicates that the corresponding column should be in a different font from the default font (usually Roman). All font names are one or two letters; a one-letter font name should be separated from whatever follows by a space or tab. The single letters *B*, *b*, *I*, and *i* are shorter synonyms for *fB* and *fI*. Font change commands given with the table entries override these specifications.

Point size changes

A key-letter followed by `p` or `P` and a number specifies the point size of corresponding table entries. If the number is a signed digit, it is taken as an increment or decrement from the current point size. If point size and column separation are both specified, one or more blanks must separate the two parameters.

Vertical spacing changes

A key-letter followed by *v* or *V* and a number specifies the vertical line spacing to be used within a multi-line corresponding table entry. If the number is a signed digit, it is taken as an increment or decrement from the current vertical spacing. If column separation and vertical spacing are both specified, the two parameters must be separated by one or more blanks or by some other specification. This request has no effect unless the corresponding table entry is a text block (see below).

Column width indication

A key-letter followed by the letter *w* or *W* and a width value in parentheses specifies minimum column width. If the largest element in the column is not as wide as the width value given after the *w*, the specified width is used. If the largest element in the column is wider than the specified value, the element width is used. Column width is also used as a default line length for included text blocks. Normal *troff* units can be used to scale the width value (if units are not specified, the default is *ens*. If the width specification is a unitless integer, the parentheses can be omitted. If the column width specification is changed during table construction, the **last** value encountered determines the result.

Equal-width columns

A key-letter followed by the letter *e* or *E* indicates equal-width columns. All columns whose key-letters are followed by *e* or *E* are made the same width, permitting the user to get a group of regularly spaced columns.

Staggered columns

A key-letter followed by *u* or *U* indicates that the corresponding entry is to be moved up one-half line. This makes it easy to have a column of differences between numbers in an adjoining column. The *allbox* option does not work with staggered columns.

Zero-width item

A key-letter followed by *z* or *Z* indicates that the corresponding data item is to be ignored in calculating column widths. This is sometimes useful when allowing headings to run across adjacent columns where spanned headings would be inappropriate.

Default

Column descriptors missing from the end of a format line are assumed to be *L* and the longest line in the format section defines the number of columns in the table. Data that would normally appear in extra columns is ignored.

The chronological order of the above features is immaterial; they need not be separated by spaces (except where indicated to avoid ambiguities involving point size and font changes). As an example, a numerical column entry in italic font and 12 point type with a minimum width of 2.5 inches and separated by 6 ens from the next column could be specified as:

```
np12w(2.5i)fI 6
```

Data to be Printed

Data for the table follows the format commands. Normally, each table input line (one or more column entries) is typed as one line of data. Very long input lines can be broken by placing a backslash character (\) at the end of the line (when successive lines are combined, the \ vanishes). Data items for successive columns on the same line are separated by tabs, or by whatever character has been specified in the *tab* global option. Here are some special cases of interest:

Troff commands within tables

An input line beginning with a dot (.) followed by anything but a number is assumed to be a command to *troff* and is passed through unchanged, retaining its position in the table. For example, space within a table can be produced by .sp requests in the data.

Full-width horizontal lines

An input line containing only the character underscore (_) or equals sign (=) results in a single or double line, respectively, extending the full width of the table.

Single-column horizontal lines

An input table entry containing only the character underscore (_) or equal sign (=) results in a single or double line extending the full width of the column. Such lines are extended to meet horizontal or vertical lines adjoining the column. To obtain these characters explicitly in a column, either precede them by \& or follow them by a space before the usual tab or newline character.

Short horizontal lines

An input table entry containing only the string _ results in a single line as wide as the contents of the column. It is not extended to meet adjoining lines.

Repeated characters

An input table entry containing only a string of the form \R<x> (where <x> is any character) is replaced by repetitions of the character <x> as wide as the data in the column. The sequence of <x>s is not extended to meet adjoining columns.

Vertically spanned items

An input table entry containing only the character string \^ indicates that the table entry immediately above spans downward over this row. It is equivalent to a table format key-letter ^.

Text blocks

To include a block of text as a table entry, precede it by `T{` and follow it by `T}`. Thus the sequence

```
. . . T{
block of
text
T} . . .
```

places, as a single entry in the table, something that cannot be conveniently typed as a simple string between tabs. Note that the `T}` end delimiter must begin a line, but additional columns of data can follow after a tab on the same line.

If more than twenty or thirty text blocks are used in a table, various limits in *troff* are likely to be exceeded, producing diagnostics such as "too many string/macro names" or "too many number registers".

Text blocks are removed from the table, processed separately by *troff*, then replaced in the table as a solid block. If no line length is specified in the block of text itself or in the table format, the default is calculated using:

$$L \times C \div (N+1)$$

where L is the current line length, C is the number of table columns spanned by the text, and N is the total number of columns in the table.

Other parameters (point size, font, etc.) used in setting the block of text are:

a) Those in effect at the beginning of the table (including the effect of the *.TS* macro),

b) Any table format specifications of size, spacing and font that use *p*, *v*, and *f* modifiers to the column key-letters, and

c) *Troff* requests within the text block itself are also recognized. However, *troff* commands within the table data but not within the text block do not affect that block.

Although any number of lines may be present in a table, only the first 200 lines are used in calculating the widths of the various columns. A multi-page table can be arranged as several single-page tables if this proves to be a problem.

Other difficulties with formatting may arise because in the calculation of column widths, all table entries are assumed to be in the font and size being used when the *.TS* command was encountered, except for font and size changes indicated (a) in the table format section and (b) within the table data (as in the entry `\s+3 data\fP\s0`). *Troff* requests can be sprinkled in a table, but care must be taken to avoid confusing the width calculations. Once the table structure is started, it is not possible to change the number of columns, space between columns, global options such as *box*, or to set the columns equal in width.

Additional Command Lines

To change table format after many similar lines (as with sub-headings or summarizations) use the *.T&* (table continue) command to alter column parameters. Here is an outline of the general technique:

```
.TS
options;
format.
data
. . .
.T&
format.
data
.T&
format.
data
.TE
```

This technique enables you to keep each table line close to its corresponding format line.

Examples

The following example tables show various techniques and command options. While each table is structured for illustrating a particular concept, other options are also illustrated such as font changes, for example. The ◊ symbol in the following examples indicates the tab character.

Table Using "box" Option

Input:

```
.TS
box;
ccc
lll.
Language◊ Authors◊ Runs on
.sp
Fortran◊ Many◊ Almost anything
Pl/1◊ IBM◊ 360/370
C◊ AT&T◊ 11/45,H6000,370
BLISS◊ Carnegie-Mellon◊ PDP-10,11
IDS◊ Honeywell◊ H6000
Pascal◊ StanfordM370
.TE
```

Output:

Language	Authors	Runs on
Fortran	Many	Almost anything
PL/1	IBM	360/370
C	AT&T	11/45,H6000,370
BLISS	Carnegie-Mellon	PDP-10,11
IDS	Honeywell	H6000
Pascal	Stanford	370

Table Using "allbox" Option

Input:

```
.TS
allbox;
c s s
c c c
n n n.
Common Stock
Year◊ Price◊ Dividend
1971◊ 41-54◊ $2.60
2◊ 46-55◊ 2.70
3◊ 46-55◊ 2.87
4◊ 40-53◊ 3.24
5◊ 45-52◊ 3.40
6◊ 51-59◊ .95*
.TE
*(first quarter only)
```

Output:

Common Stock		
Year	Price	Dividend
1971	41-54	$2.60
2	46-55	2.70
3	46-55	2.87
4	40-53	3.24
5	45-52	3.40
6	51-59	.95*

*(first quarter only)

Table Using "vertical bar" Key-letter Feature

Input:

```
.TS
box;
c s s
c| c| c
l| l| n.
Major New York Bridges
_
Bridge◊ Designer◊ Length
_
Brooklyn◊ J.A. Roebling◊ 1595
Manhattan◊ G. Lindenthal◊ 1470
Williamsburg◊ L.L. Buck◊ 1600
_
Queensborough◊ Palmer &◊ 1182
◊ Hornbostel
_
◊ ◊ 1380
Triborough◊ O.H. Ammann◊ _
◊ ◊ 383
_
Bronx Whitestone◊ O.H. Ammann◊ 2300
Throgs Neck◊ O.H. Ammann◊ 1800
.TE
```

Output:

Major New York Bridges		
Bridge	Designer	Length
Brooklyn Manhattan Williamsburg	J.A. Roebling G.Lindenthal L.L. Buck	1595 1470 1600
Queensborough	Palmer & Hornbostel	1182
Triborough	O.H. Ammann	1380 383
Bronx Whitestone Throgs Neck	O.H. Ammann O.H. Ammann	2300 1800

Table Using Horizontal Lines in Place of Key-letters

Input:

```
.TS
box;
L L L
L L _
L L| LB
L L _
L L L.
january⇄ february⇄ march
 april⇄ may
june⇄ july⇄ Months
august⇄ september
october⇄ november⇄ december
.TE
```

Output:

january	february	march
april	may	
june	july	**Months**
august	september	
october	november	december

Table Using Additional Command Lines

Input:

```
.TS
box;
cfB s s s.
Composition of Foods
_
.T&
c |c s s
c | c s s
c | c | c | c
Food◊ Percent by Weight
\^◊ _
\^◊ Protein◊ Fat◊ Carbo-
\^◊ \^◊\^◊ hydrate
_
.T&
l | n | n | n.
Apples◊ .4◊ .5◊ 13.0
Halibut◊ 18.4◊ 5.2◊ . . .
Lima Beans◊ 7.5◊ .8◊ 22.0
Milk◊ 3.3◊ 4.0◊ 5.0
Mushrooms◊ 3.5◊ .4◊ 6.0
Rye Bread◊ 9.0◊ .6◊ 52.7
.TE
```

Output:

Composition of Foods			
Food	Percent by Weight		
	Protein	Fat	Carbo-hydrate
Apples	.4	.5	13.0
Halibut	18.4	5.2	. . .λ
Lima Beans	7.5	.8	22.0
Milk	3.3	4.0	5.0
Mushrooms	3.5	.4	6.0
Rye bread	9.0	.6	52.7

Table Using Text Blocks

Input:

```
.TS
allbox;
cfI s s
c cw(2i) cw(2i)
l l l.
New York Area Rocks
.sp
Era\t Formation\t Age (years)
Precambrian\t Reading Prong\t >1 billion
Paleozoic\t Manhattan Prong\t 400 million
Mesozoic\tT{
.na
Newark Basin, incl.
 Stockton, Lockatong, and Brunswick
 formations
.ad
T}\t 200 million
Cenozoic\t Coastal Plain\t T{
.na
On Long Island 30,000 years;
Cretaceous sediments redeposited
by recent glaciation
.ad
T}
.TE
```

Output:

New York Area Rocks		
Era	Formation	Age (years)
Precambrian	Reading Prong	>1 billion
Paleozoic	Manhattan Prong	400 million
Mesozoic	Newark Basin, incl. Stockton, Lockatong, and Brunswick Formations	200 million
Cenozoic	Coastal Plain	On Long Island 30,000 years; Cretaceous sediments redeposited by recent glaciation

MANUAL COMMENT CARD

Text Formatters
HP-UX Concepts and Tutorials

HP Part Number 97089-90032 August 1986

Please help us improve this manual. Circle the numbers in the following statement that best indicate how useful you found this manual. Then add any further comments in the spaces below. **In appreciation of your time, we will enter your name in a quarterly drawing for an HP calculator.** Thank you.

The information in this manual:

Is poorly organized	1	2	3	4	5	Is well organized
Is hard to find	1	2	3	4	5	Is easy to find
Doesn't cover enough	1	2	3	4	5	Covers everything
Has too many errors	1	2	3	4	5	Is very accurate

Particular pages with errors? _____

_____

Comments: _____

_____

_____

_____

_____

_____

_____

Name: _____

Job Title: _____

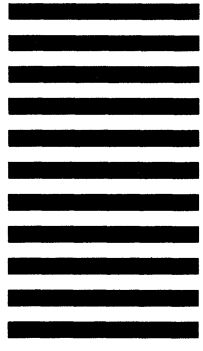
Company: _____

Address: _____

_____

☐ Check here if you wish a reply.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



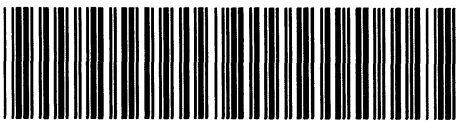
BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 37 LOVELAND, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

Hewlett-Packard Company
Fort Collins Systems Division
Attn: Customer Documentation
3404 East Harmony Road
Fort Collins, Colorado 80525

HP Part Number
97089-90032
Microfiche No. 97089-99032
Printed in U.S.A. 8/86



For Internal Use Only