1032

Bell Laboratories **Cover Sheet for Technical Memorandum**

The information contained herein is for the use of employees of Bell Laboratories and is not for publication. (See GEI 13.9-3)

Title- **TROFF Users' Manual**

Date- **April 19, 1974**

TM- **74-1271-4**

Other Keywords- **Text Formatting**
Typesetting

Author	Location	Extension	Charging Case- 39199
J. F. Ossanna	**MH-2C-521**	**3520**	Filing Case- **39199-11**

ABSTRACT

TROFF is a text formatter which produces output for a Graphic Systems Phototypesetter and which is available on Center 127's UNIX Time-Sharing System. This User's Manual consists of: a description of usage; a Request Summary and Index; a Reference Manual keyed to the index; and a set of Tutorial Examples. TROFF accepts lines of text interspersed with lines of format control information and formats the text into a printable, paginated document having a user-designed style. TROFF offers unusual freedom in document styling including: arbitrary style headers and footers; arbitrary style footnotes; multiple automatic sequence numbering for paragraphs, sections, etc; multiple column output, dynamic font and point-size control, arbitrary horizontal and vertical local motions at any point, and a family of automatic overstriking, bracket construction, and line drawing functions. Basic mechanisms include: macros and strings; macro interpolation at vertical position traps; number registers; multiple text processing environments; conditional input; and an insert ability.

UNED (AH)

Pages Text ~~X~~ 2	Other ~~X~~ 29	Total 31
No. Figures 0	No. Tables 2	No. Refs. 4

SEE REVERSE SIDE FOR DISTRIBUTION LIST

Bell Laboratories

Subject- **TROFF Users' Manual** - Case 39199-11

Date- **April 19, 1974**

From- **J. F. Ossanna**

TM- **74-1271-4**

MEMORANDUM FOR FILE

Introduction

TROFF is a text processor on the PDP-11/45 UNIX Time-Sharing System that formats text for a Graphic Systems phototypesetter. It accepts lines of text interspersed with lines of format control information (request lines) and formats the text into a printable, paginated document having a user-designed style. The request syntax, many of the requests, and the general method of use is similar to that of NROFF[1]. TROFF offers unusual freedom in document styling including: arbitrary style headers and footers; arbitrary style footnotes; multiple automatic sequence numbering for paragraphs, sections, etc; multiple column output, dynamic font and point-size control, arbitrary horizontal and vertical local motions at any point, and a family of automatic overstriking, bracket construction, and line drawing functions.

Usage

The general form of invoking TROFF at UNIX command level is

troff options files

where "options" represents any of a number of optional arguments and "files" represents the list of files containing the document to be formatted. The options, which may appear in any order so long as they appear before the files, are:

Option	*Effect*
+N	Commence output at the first page whose page number is N (independent of whether or not the page number is being printed).
−N	End output after the first page whose page number is N.
−nN	Number first generated page N.
−t	Direct output to the standard output instead of the phototypesetter.
−f	Refrain from feeding out paper and stopping phototypesetter at the end of the run.
−w	Wait until phototypesetter is available, if currently busy.
−i	Read standard input after the input files are exhausted. TROFF automatically reads the standard input, if no input files are given.
−a	Send a printable approximation of the results to the standard output.
−pN	Print all characters in point size N while retaining all prescribed spacings and motions.
−mx	Simulates an ".so /usr/lib/tmac.x" request.

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

troff +7 −w file1 file2

requests formatting of a document contained in the files named "file1" and "file2", beginning with page 7, and waiting, if necessary, for the phototypesetter to become available.

If no files are given, the input is taken from the standard input, which may be a "pipe"[2]. For example, TROFF may be used with EQN[3], the equation preprocessor, as follows:

eqn files | troff options

The "|" indicates the piping of EQN's output to TROFF's input.

The remainder of this manual consists of: a Request Summary and Index; a Reference Manual keyed to the index; and a set of Tutorial Examples. Additional tutorial examples may be found in [4].

J. F. Ossanna

MH-1271-

References

[1] J. F. Ossanna, *NROFF User's Manual*, MM-73-1271-2, (February 5, 1973).

[2] K. Thompson, D. M. Ritchie, *UNIX Programmer's Manual*, Fourth Edition (November 1973).

[3] B. W. Kernighan, L. L. Cherry, *Typesetting Mathematics — User's Guide*, TM-74-1273-3, (March 1974).

[4] B. W. Kernighan, *TROFF Made Trivial*, TM-73-1273-10, (November 1973).

Att:
pages 3-31

REQUEST SUMMARY AND INDEX

Request Form	*Initial Value*	*If no Argument*	*Cause Break**	*Explanation*
I. Font and Character Size Control				
.ps ±N†	10 point	prev	no	Point size (also \s±N).
.ss N†	12/36 Em	ignored	no	Space character size set to N/36 Em.
.cs F N† M†	off	-	no	Constant character space (width) mode.
.ft F	Roman	prev	no	Font change (also \fF); F=R,I,B,S,G,C,or P.
.fp N F	R,I,B,S	ignored	no	Physical font position specification.
.cp N	off	off	no	Constant output point-size mode.
II. Page Control				
.pl ±N	11 in	11 in	no	Page length.
.bp ±N	N=1	-	yes	Begin page.
.pn ±N	N=1	ignored	no	Page number.
.po ±N	26/27 in	prev	no	Page offset.
.ne N	-	N=1 VS	no	Need N vertical space (VS = vertical spacing).
.mk a	none	internal	no	Mark current vertical place.
.rt −N	none	internal	no	Return to marked vertical place.
III. Text Filling, Adjusting, and Centering				
.br	-	-	yes	Break.
.fi	fill	-	yes	Fill output lines.
.nf	fill	-	yes	No filling and adjusting.
.ad c	adj,norm	adjust	no	Adjust mode on.
.na	adjust	-	no	No adjusting.
.ce N	off	N=1	yes	Center N input text lines.
IV. Vertical Spacing				
.vs N‡	12points	prev	no	Vertical line spacing (VS).
.sp N	-	N=1 VS	yes	Space vertically distance N.
.lv N	-	N=1 VS	no	See "sv" below.
.sv N	-	N=1 VS	no	Save vertical distance N.
.os	-	-	no	Output saved vertical distance.
.ns	space	-	no	No-space mode on.
.rs	-	-	no	Restore spacing.
V. Line Length and Indenting				
.ll ±N	6.5 in	prev	no	Line length.
.in ±N	N=0	prev	yes	Indent.
.ti ±N	-	ignored	yes	Temporary indent.
VI. Macros, Diversion, and Line Traps				
.de xx	-	ignored	no	Define or redefine a macro.
.am xx	-	ignored	no	Append to a macro.
.ds xx	-	ignored	no	Define a string.
.as xx	-	ignored	no	Append to a string.
.rm xx	-	ignored	no	Remove macro or string.
.di xx	-	end	no	Divert output to macro "xx".

*The use of "'" as control character (instead of ".") suppresses the break function.

†Scale indicators ignored.

‡Default scaling is points.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.da xx	-	end	no	Divert and append to "xx".
.wh −N xx	-	-	no	When; set a line trap.
.ch −N−M	-	-	no	Change trap location.
.ch xx −M	-	-	no	"
.dt N xx	-	off	no	Set a diversion trap.
.it N xx	-	off	no	Set an input-line count trap.
VII. Number Registers				
.nr a ±N −M		-	no	Number register.
.nr ab ±N −M		-	no	"
.nc c	\n	\n	no	Number character.
.ar	arabic	-	no	Arabic numbers.
.ro	arabic	-	no	Lower case roman numbers.
.RO	arabic	-	no	Upper case roman numbers.
VIII. Tabs, Leaders, and Fields				
.ta N,...	.5,1,...in	none	no	Tab settings.
.tc c	none	none	no	Tab replacement character.
.lc c	.	.	no	Leader replacement character.
.fc a b	off	off	no	Set field delimiter and pad characters.
IX. Input and Output Conventions and Character Translations				
.ec c	\	\	no	Set escape character.
.lg N	on	on	no	Ligature mode.
.cc c	.	.	no	Basic control character.
.c2 c	'	'	no	Nobreak control character.
.li N	-	N=1	no	Accept input lines literally.
.tr abcd....	none	-	no	Translate on output.
X. Local Horizontal and Vertical Motions, and the Width Function				
XI. Overstrike, Bracket, Line-drawing, and Zero-width Functions				
XII. Hyphenation.				
.nh	hyphenate	-	no	No hyphenation.
.hy	hyphenate	-	no	Hyphenate.
.hc c	none	none	no	Hyphenation indicator character.
XIII. Three Part Titles.				
.tl 'left'center'right'		-	no	Title.
.lt N	6.5 in	prev	no	Length of title.
XIV. Output Line Numbering.				
.nm ±N M S I		off	no	Number mode on or off, set parameters.
.np M S I	none	reset	no	Number parameters set or reset.
XV. Conditional Input Line Acceptance				
.if c anything		-	no	If condition true accept "anything".
.if !c anything		-	no	"
.if N anything		-	no	"
.if !N anything		-	no	"

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
XVI. Environment Switching.				
.ev N	N=0	prev	no	Environment pushed down.
XVII. Insertions from the Standard Input Stream				
.rd prompt	-	bell	no	Read insert.
.ex	-	-	no	Exit.
XVIII. Input File Switching				
.so filename		-	no	Switch source file (push down).
.nx filename		-	no	Next file.
XIX. Miscellaneous				
.tm string	-	-	no	Teletype message.
.hs N	on	on	no	Control high-speed multiple scan mode (N=0 for off).
.ig	-	-	no	Ignore.
.fl	-	-	yes	Flush output buffer.
.ab	-	-	no	Abort.
XX. Error Messages				

Escape Sequences for Characters, Indicators, and Functions

Ref.	*Input*	*Meaning*
-	\\	\ (to prevent or delay the interpretation of \)
-	\e	Directly printable version of the current escape character.
-	\'	' (equivalent to \(aa)
-	\`	` (equivalent to \(ga)
-	\-	− (equivalent to \(mi)
-	_	_ (equivalent to \(ul)
-	\\|	1/6 Em space character
IX.*VI*	\&	Non-printing, zero width character
IX.*IX*	\!	Transparent line indicator
VI.*III*	\$	Argument indicator
IX.*IV*	\(	Character name indicator
VI	*	String indicator
-	\:	Generates ASCII ETX (003) for post processor use
VIII	\a	Non-interpreted leader character
XI.*III*	\b	Bracket building function
IX.*II*	\c	Interrupt text processing
X.*I*	\d	Forward (down) 1/2 Em vertical motion
I	\f	Font change function
X.*I*	\h	Generalized local horizontal motion function
X.*III*	\k	Mark horizontal input place
XI.*IV*	\l	Line drawing function
VII	\n	Number register indicator
XI.*I*	\o	Overstrike function
III	\p	Break and spread output line
X.*I*	\r	Reverse 1 Em vertical motion
I	\s	Point-size change function
VIII	\t	Non-interpreted horizontal tab
X.*I*	\u	Reverse (up) 1/2 Em vertical motion

X.*I* \v Generalized local vertical motion function
X.*II* \w Width function
IV.*II* \x Extra line-space function
XI.*II* \z Zero width character function
IX.*II* \(newline) Concealed (ignored) newline

REFERENCE MANUAL

Explanation

Control input lines. Input lines beginning with a "control" character (ordinarily "." or "'") are interpreted as containing either format control "requests" or as specifying the interpolation of a user defined macro. Such lines which contain neither a recognizable request name nor a defined macro name are ignored. The "break" function—the forced output of a partially filled line—associated with any request can be suppressed by using the no-break control character (ordinarily "'" instead of ".") to indicate control lines.

Phototypesetter Units. The Graphic Systems phototypesetter has a horizontal resolution (along the line) of 432 increments per inch and a vertical resolution of 144 increments per inch. TROFF internally uses 432 basic units per inch for both directions. TROFF accepts parameters specified in several different units and stores them internally with full resolution. Some relationships involving typesetting units are:

72 Points = 1 Inch
6 Picas = 1 Inch
1 Em = (Point-size) Points
1 En = 1/2 Em

Numerical parameter input. Numerical parameters are scaled by TROFF according to a scale indicator following (and attached to) the number.

Form	*Meaning*	*Parameter stored*
Ni	N Inches	N*432 units
Nc	N Cm.	N*170 units
Np	N Points	N*6 units
Nu	N Units	N units
Nm	N Ems	N*6*(point-size) units
Nn	N Ens	N*3*(point-size) units
N	N Default units (see below)	

The default unit for horizontally-oriented requests (ll, in, ti, ta, lt, po) is Ems; the default unit for vertically-oriented requests (pl, wh, ch, sp, sv, ne, rt) is vertical-spacing units (see "vs" request below). Numerical arguments are indicated below in several symbolic forms: ±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 decrement it by N respectively; −N means that the argument may take the form N or −N and that the effect is to set the parameter to N or −N. Other capital letters are used for additional numerical arguments. If N is immediately preceded by "|", the number is evaluated as the distance to the horizontal/vertical place specified. Vertically-oriented requests refer to vertical places on the page; *all* other requests refer to horizontal places on the *input* line. For example, ".sp |1i" means space vertically to 1 inch from the top. The number, N, may be specified in decimal-fraction form but the parameter finally stored is truncated to an integer. Number arithmetic may be used; care must be taken to attach the correct scale indicator to *each* number involved. For example, ".ll 3.25i", ".ll 3i+18p", ".ll 1404u", and ".ll 6.5i/2u" all specify the same line length. When number arithmetic is used in the ±N case, the entire N is evaluated before the result is taken as an increment (e. g. −1i−6p will decrement by (1i−6p)). Certain requests that set a parameter value will restore the previously set value, if no new value is specified; only the most recent previous value is saved.

Character string arguments. Single character arguments are indicated by single lower case letters. Character string arguments are indicated by multi-character mnemonics. The space character shown immediately after the two character requests can be omitted in those cases where the first argument is numeric.

I. Font and Character Size Control

I. Fonts. Six fonts are available (known to TROFF): Times Roman; Times Italic; Times Bold; Geneva Regular; News Gothic Condensed; and a Special Mathematical Font. The Roman, Italic, Bold, and Special fonts are normally mounted and are represented throughout this document. Their complete character sets are shown in Table I. If the Special Font is mounted, all ASCII graphical characters may be

printed; certain input ASCII characters are, however, mapped as follows:

Character	*Printed as*
` (grave)	‘ (open single quote)
´ (acute)	’ (close single quote)
– (minus)	- (hyphen)
_ (underline)	— (dash)

The characters ´, `, _, and – (minus) may be input by means of their names shown in Table II or by escape conventions given in §IX.*I*. The ASCII characters ", ´, `, <, >, \, {, }, ~, ^, and _ exist only on the Special Font and are printed as a 1-Em space if that Font is not mounted. A number of non-ASCII characters are available in the Times Roman, Italic, Bold, and Geneva Fonts and are shown in Table II.

The current font is changed using the "ft" request, or may be changed at *any* point by imbedding a \fF at the point where "F" refers to the desired font.

II. Character size. Point sizes available are 6, 7, 8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24, 28, and 36. This is a range of 1/12 inch to 1/2 inch. Requested point size values that are between two valid sizes yield the larger of the two. The "ps" request is used to change or restore the point size; or the point size may be changed between any two characters by imbedding a \sN (or\s±N where $0 \leqslant N \leqslant 9$) at the desired point; if N is zero in the latter case, the previous size is restored.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.ps ±N†	10 point	prev	no	Point size (also may imbed \s±N). Any positive size value may be requested; if invalid, the next larger valid size will result, with a maximum of 36. A paired sequence of +N...–N values will work inasmuch as the requested values are remembered.
.ss N†	12/36 Em	ignored	no	Space size. The size of the space character is set to N/36 Ems. This size is the minimum word spacing in adjusted text.
.cs F N† M†	off	-	no	Constant character space (width) mode is set on for font F (if mounted) and the width of every character is taken to be N/36 Ems. If M is absent, the Em is that of the character's point size; if M is given, the Em is M-points. All affected characters are centered in this space, including those with a actual width larger than this space. Special Font characters occurring while the current font is F are also so treated. If N is absent, the mode is turned off.
.ft F	Roman	prev	no	Font change (may also imbed \fF). F = R, I, B, G, C, P represent Roman, Italic, Bold, Geneva, Condensed, and the Previous font respectively. F = S is ignored because as the Special Font is automatically employed for characters contained in it.
.fp N F	R,I,B,S	-	no	Font position. The Phototypesetter has four fonts physically mounted. Each font consists of a film strip which can be mounted on a numbered quadrant of a wheel. F (=R, I, B, G, C, or S) is the font which is mounted on font wheel position N. The default mounting sequence assumed by TROFF is R, I, B, and S on positions 1, 2, 3 and 4.

†Scale indicators ignored.

.cp N	off	off	no	Constant output point-size is set to N-points (same as the −pN option). This mode provides the fastest production of documents containg many point-size changes. Text processing is done normally and all prescribed spacings and motions are produced.

II. 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 (from the bottom). See § VI and Tutorial Examples.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.pl ±N	11 in	11 in	no	Page length set to ±N.
.bp ±N	N=1	-	yes	Begin page. The current page is ejected and a new page is begun. If ±N is given, the new page number will be ±N. See request "ns".
.pn ±N	N=1	ignored	no	Page number. The next page (when it occurs) will have the page number ±N. A "pn" occurring before the first break or first text will set the page number for the first page.
.po ±N	26/27 in	prev	no	Page offset. The current "left margin" is set to ±N. Maximum (Line-length)+(Page-offset) is about 7.54 inches. See § V.
.ne N	-	N=1 VS	no	Need a vertical distance N. If the distance, D, to the next trap position is less than N, the paper is moved forward the distance D, which will spring the trap. If there are no remaining traps on the page, D will be the distance to the bottom of the page. See § VI.
.mk a	none	internal	no	Mark. The current vertical place on the page is stored for future reference. If "a" is given, the place is stored in number register "a". (see "rt" request.)
.rt −N	none	internal	no	Return to vertical place. The paper will be moved in the *reverse direction only* to a place a distance N from the top of the page or, if N is negative, to a place a distance N before the current place, or, if N is absent, to a previously marked place (see request "mk"), if any.

III. Text Filling, Adjusting, and Centering

The default fill mode is to fill output lines; input words are taken from the next input line or output words are deferred until the next output line to produce output lines that are full but within the current line length size. The default adjust mode is to adjust lines for a uniform right (as well as left) margin; if a fully formed line contains fewer character positions than the current line length, the blank spaces between words are expanded to achieve the current line length. Both of these processes may be turned off. During filling hyphenation is automatically attempted when the next word does not fit on the line; this process may be turned off also. (See § XII).

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.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.br	-	-	yes	Break. The filling of the line currently being filled out is stopped and the line is printed. Text lines beginning with spaces and empty text lines (blank lines) cause a break.
.fi	fill	-	yes	Fill output lines. Subsequent output lines are filled.
.nf	fill	-	yes	Nofill. Subsequent output lines are neither filled nor adjusted. Input text lines are copied directly to output lines without regard for the current maximum line length (see request "ll").
.ad c	adj,norm	adjust	no	Adjust mode is turned on. If fill mode is not on, adjustment will be deferred until fill mode is back on. If the adjustment type indicator character, "c", is present the adjustment type is changed; if "c" is "n", the normal (default) is restored — both margins will be adjusted uniform; if "c" is "r", only the right margin is adjusted — the left margin will be ragged; if "c" is "c", the line is centered — both margins will be ragged.
.na	adjust	-	no	Noadjust. Adjustment is turned off; i.e. the left margin will be uniform and the right margin will be ragged. The adjustment type is not changed. Output line filling will still occur if fill mode is on.
.ce N	off	N=1	yes	Center the next N input text lines within the current line length. A break automatically occurs after each line. If the input line is longer than the line length, it will be left adjusted. If N=0, any residual centering count is cleared.

IV. Vertical Spacing

I. Line spacing. The vertical line spacing (VS) may be dynamically set with a resolution of 1/144 inch. The VS should be set large enough to accommodate the character size of the affected output lines. In the region of typical point sizes (9-12 points) the usual practice is to set the VS to 2 points greater than the point size; the default is 10-point type on a 12-point spacing (as in this document). The VS is specified by the "vs" request and affects the next output line. The VS used for a particular line can be qualified by means of the extra space function (see next).

II. Extra Line-Space. Provision for extra vertical space both before and after some particular output line can be obtained by including the function \x'N' within or attached to a word. 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 appear in the same line, the maximum value is used.

III. Blocks of vertical space. Vertical space is ordinarily requested using "sp" which honors the no-space mode and which does not space past a trap. When a contiguous vertical space must be reserved the "sv" request should be used.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.vs N‡	12 points	prev	no	Vertical spacing size (VS). Also see discussion of the "extra-space" function below.
.sp N	-	1 VS	yes	Space vertically in either direction. If N is negative, the motion is upward (backward). Downward (forward) motion is limited to the distance to the nearest trap. Reverse motion

‡Default scaling is points.

				is limited to the distance to the top of the page. If the no-space mode is on, no spacing occurs.
.lv N	-	N=1	no	See "sv" request next.
.sv N	-	N=1	no	Save N vertical space. 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 (see request "os"). Subsequent "sv" requests will overwrite any still remembered N.
.os	-	-	no	Output saved vertical space. No-space mode has no effect. Used to finally output a block of vertical space requested by the "sv" request.
.ns	space	-	no	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 is produced.
.rs	space	-	no	Restore spacing. The no-space mode is turned off.
Blank text line.		-	yes	Causes a break and output of a blank line exactly like "sp 1".

V. Line Length and Indenting

Requests are provided to set and reset the line length and the extent of indent. The line length includes any indent spaces but does not include page offset spaces. As long as fill mode is on, the length of text on an output line is less than or equal to the line length minus the indent (see § II).

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.ll ±N	6.5 inches	prev	no	Line Length is set to ±N. Maximum (Line-length)+(Page-offset) is about 7.54 inches.
.in ±N	N=0	prev	yes	Indent is set to ±N. If the line length is L and the indent is N, N horizontal space is put out at the beginning of each output line and the text on the remainder of the line is constrained to a size L−N.
.ti ±N	-	N=1	yes	Temporary indent. The next output text line will be indented a distance ±N (±N with respect to the current indent). The resulting temporary indent may not be negative. The current indent is not changed.

VI. Macros, Strings, Diversion, and Line Traps

I. Macros and strings. A macro is a named set of one or more lines that may be invoked by name or by having reached a specified vertical position on the page. Macro names are one or two characters long and may usurp previously defined request names or macro names. Macros are defined (or redefined) by the "de" request or by output diversion (see "di"). Existing macros can be appended to using the "am" and "da" requests. A macro named "xy" may be invoked by an input line beginning with ".xy"; the rest of the line may contain up to nine argument strings. In addition, a "trap" may be set at a vertical place on the page to invoke the macro (see "wh"). Macros may contain arbitrary request and text lines. A "string" is a macro containing a line fragment (without a terminal new-line). Strings are created using the "ds" or "as" requests. They are then interpolated into the input at any point by the sequence "*x" or "*(xx" where "x" and "xx" are one and two character string names respectively.

II. Copy mode input interpretation. During macro and string definition (or appending to) the input is processed in a "copy" mode. The only input processing that occurs is escape mapping (see §IX.*I*) and: (1) number registers indicated by "\n" are expanded; (2) string references (indicated by "*") are interpolated into the input; and (3) argument references (indicated by "\$") are interpolated into the input. In cases

(1-3) the interpretation can be suppressed by prepending a "\". 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.

The concealed newline (\(newline)) is always processed (thrown away). Tabs, leaders, backspaces, and fields are not processed in copy mode. Except for the interpretation of number registers, strings, and arguments, functions are not interpreted or performed.

III. Arguments. When a macro is invoked as a request the request line may contain up to nine arguments separated by blanks. If the desired arguments won't fit on a line, a concealed new-line may be used to continue on the next line (see § IX.*1*). If an argument contains blanks, it must be surrounded by double-quotes. For example,

```
.xx arg1 "a r g 2" arg3
```

calls macro "xx" with three arguments. Each time a macro is invoked any arguments available at that level are pushed down and any new arguments are made available. No arguments are available at the top (input file) level. The arguments available at the current level are invoked (i. e. included in the current input) by

```
\$N
```

where \$ is the argument indicator and N is an digit from 1 to 9. If the invoked argument doesn't exist, a null string is included. If a macro is to contain "\$N", it is necessary to conceal the "\$" when the macro definition is being copied; "\\$N" would copy as "\$N". For example, the macro "xx" may defined by

```
.de xx
Today is \\$1 the \\$2\\$3.
..
```

and called by

```
.xx Monday 14 th
```

to produce the text

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

IV. Diversions. Processed output may be diverted into a macro for purposes such as footnote processing (see Tutorial §V) or determining the vertical size of some text for conditional changing of pages or columns (number register "dn" contains the size of the last diverted text). Processed text that is diverted into a macro retains its vertical dimensions when reread in "nofill" mode regardless of the current vertical line-space. During a diversion the internal extra-line-space parameters are saved and restored.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.de xx	-	ignored	no	Define or redefine a macro with the one or two character name "xx". The contents of the macro begins on the next input line. Input lines are copied until the definition is terminated by a line beginning with "..". A macro may contain "de" requests provided the corresponding ".." is concealed to prevent copy termination; "\\.." will copy as "\.." and be reread as ".."
.am aa	-	-	no	Append to macro (append version of "de").
.ds aa string		-	no	Define string. The character string "string" (without a terminal new-line character) is defined as a macro having the (one or two character) name "aa". An initial double-quote, "", is stripped off to permit initial blanks. This request is like the "de" request, except that the resulting macro consists of only "string".

.as aa string	-		no	Append to string (append version of "ds").
.rm xx	-	-	no	Remove macro or string. The macro or string named "xx" is removed from the name list. Subsequent references will have no effect.
.di xx	-	end	no	Divert output into the macro named "xx". The macro name "xx" is (re)defined at this point. 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. Diversions cannot be nested. The vertical size of the diverted output is kept in a number register (named "dn") for possible later use.
.da aa	-	end	no	Append to diversion (append version of "di").
.wh −N xx	-	-	no	When the page is filled to or beyond the vertical place −N invoke macro "xx". Any macro previously planted at vertical place −N is replaced by "xx". A positive N is measured from the top of the page and −N is measured from the bottom. A zero N will set a trap at the beginning of a page and should be used for headers. If no macro name is given, the trap planted at line −N, if any, is removed.
.ch −N −M	-	-	no	See next.
.ch xx −M	-	-	no	Change the trap planted at vertical place −N to occur instead at vertical place −M. Alternatively, change the place at which the macro "xx" is planted to be −M. If no trap exists at vertical place −N, the request is ignored.
.dt N xx	-	off	no	Diversion trap. A trap is set to invoke the macro "xx" at a vertical distance N during any diversion. If no arguments are given, the diversion trap is removed.
.it N xx	-	off	no	Set an input-line count trap. The macro "xx" will be invoked after N lines of text input (control or request lines don't count) have been read. The text may come from in-line text, may be interpolated by inline macros, or may occur in trap-invoked macros. Both the value of N and the name "xx" are associated with the current environment.

VII. Number Registers

It is possible to define and use number registers to automatically sequence-number sections, paragraphs, lines, etc. A number register may be used any time a number is expected. Number registers have one or two character names and are invoked by the sequences:

\na or \n+a	(one character name)
\n(ab or \n+(ab	(two character name)

where "\n" is the "number character" indicating that the next character(s), unless "+" or "(", is the name of a number register and where "a" or "ab" is the number register name. The "+" in the second example specifies that the register is to be auto-incremented prior to use. The "(" in the two character name examples indicates the presence of a two character name. When invoked the number register is converted to decimal, lower-case Roman, or upper-case Roman and interpolated into the input stream. If the number character "\n" is used within a macro definition, the number will be invoked at the time the definition is read unless the "\" is preserved by an additional preceding "\"; i. e., expressing the number character as "\\n" will delay number invocation until the macro is invoked.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.nr a ±N −M		-	no	See next.
.nr ab ±N −M		-	no	Number register definition or modification. "a" or "ab" is the register name. The register is (re)set to ±N. The increment to be used for auto-incrementing is set to −M.
.nc c	\n	\n	no	Number character is set to "c". For example, if "c" = "X", "Xa" will invoke the register "a".
.ar	arabic	-	no	Arabic numbers (see below).
.ro	arabic	-	no	Lower case roman numbers (see next).
.RO	arabic	-	no	Upper case roman numbers. Number registers will subsequently be converted into Arabic, lower case Roman, or upper case Roman respectively. This number conversion mode also applies to the page number conversion in titles (see request "tl").

Clearly, "+" and "(" cannot be used as number register names. In addition, there are reserved (internally defined) names:

Name	*Use*
%	Current page number.
yr	Last two digits of current year.
mo	Current month (1-12).
dy	Current day of the month.
dw	Current day of the week (1-7).
nl	Current vertical place on a page.
dn	Vertical size of last diversion.
hp	Current horizontal place on input line.
ct	Character type (see "width" function in §X.*II*).
.f	Current font as physical quadrant (read only).
.s	Current point size (read only).
.v	Current vertical line spacing (read only).
.p	Current page length (read only).
.o	Current page offset (read only)
.l	Current line length (read only).
.i	Current indent (read only).
.t	Distance to the next trap (read only).
.$	Number of arguments available at the current macro level (read only).
.a	Post-line extra line-space most recently requested using \x'N' (read only).
.x	Reserved version-dependent register (read only).

VIII. Tabs, Leaders, and Fields

I. Tabs and leaders. The horizontal tab character is replaced by pure horizontal motion corresponding to the distance to the next tab stop on that *input* line. Optionally, the tab may be replaced by a user-specified replacement-character string having enough replacement-characters to fill out the distance to the tab stop; any fractional character residual motion is prepended to the string. The "leader" character (ASCII "SOH") is treated exactly like the horizontal tab except that the default mode is to use "." as the replacement-character. Tabs or leaders encountered after the last stop are ignored.

II. Fields. A "field" is surrounded by a user-defined field delimiter character. A field contains a string consisting of sub-strings separated by padding indicator characters. The field length is the distance on the *input* line from the place 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 "/", and if the padding indicator is "^", /'xxx/ and /'xxx^/ represent right-

adjusted and centered "xxx" respectively.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.ta N,...	.5,1 in...	0	no	Tab settings. Default tabs are at 0.5, 1.0, 1.5, ... 6.0 inches; a total of 12 stops may be set. The stop values may be separated by commas, spaces, or any other nonnumerics. Values preceded by "+" *or those* numerically less than the previous stop position are treated as increments to the previous stop.
.tc c	none	none	no	The tab replacement-character is set to "c" or removed. In the absence of a replacement character tabs are converted into pure motion.
.lc c	. (period)	none	no	Leader replacement character is set to "c" or removed. In the absence of a replacement character tabs are converted into pure motion.
.fc a b	off	off	no	The field delimiter is set to "a"; the padding indicator character is set to "space" or to "b", if given. In the absence of arguments the field mechanism is turned off.

IX. Input and Output Conventions and Character Translations

Certain character translations and character sequence interpretations occur both when the input file is read and when stored macros and strings are invoked and reread. This section summarizes how and when these occur and to what extent they can be controlled, delayed, or suppressed.

I. Input character translations. The ASCII graphic characters represent themselves except for `, ', —, and _ which represent ‘, ’, -, and — respectively. The ASCII controls "SOH", bell, backspace, tab, newline, and escape are accepted and *all* others are ignored. The character "\" plays an important role during input by modifying the interpretation of the character following; for this reason the "\" is called the "escape" character but it should not be confused with the ASCII control character of the same name. For example \`, \', \—, _, and \\ are used to input characters that print as `, ', —, _, and \ respectively. In addition, the "\" is used to introduce various indicators, functions, and local motions (see later). A complete list of characters modified by a preceding "\" follows; the treatment of all other characters is unaffected.

Ref.	*Input*	*Meaning*
-	\\	\ (to prevent or delay the interpretation of \)
-	\e	Directly printable version of the current escape character
-	\'	' (equivalent to \(aa)
-	\`	` (equivalent to \(ga)
-	\—	— (equivalent to \(mi)
-	_	_ (equivalent to \(ul)
-	\\|	1/6 Em space character
IX.*VI*	\&	Non-printing, zero width character
IX.*IX*	\!	Transparent line indicator
VI.*III*	\$	Argument indicator
IX.*IV*	\(	Character name indicator
VI	*	String indicator
-	\:	Generates ASCII ETX (003) for post processor use
VIII	\a	Non-interpreted leader character
XI.*III*	\b	Bracket building function
IX.*II*	\c	Interrupt text processing
X.*I*	\d	Forward (down) 1/2 Em vertical motion
I	\f	Font change function
X.*I*	\h	Generalized local horizontal motion function
X.*III*	\k	Mark horizontal input place

XI.*IV*	\l	Line drawing function
VII	\n	Number register indicator
XI.*I*	\o	Overstrike function
III	\p	Break and spread output line
X.*I*	\r	Reverse 1 Em vertical motion
I	\s	Point-size change function
VIII	\t	Non-interpreted horizontal tab
X.*I*	\u	Reverse (up) 1/2 Em vertical motion
X.*I*	\v	Generalized local vertical motion function
X.*II*	\w	Width function
IV.*II*	\x	Extra line-space function
XI.*II*	\z	Zero width character function
IX.*II*	\(newline)	Concealed (ignored) newline

The escape character may be changed.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.ec c	\	\	no	Change escape character to "c" or reset it to "\".

II. Interrupted text. The text for a "nofilled" line can be interrupted by terminating the partial line with a "\c". The next encountered text will be considered to be a continuation of the same line of input. If the intervening control lines cause a break, the partial line will be forced out. Similiarly, 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 but the partial word will not unless no partial line exists.

III. Ligatures. Three ligatures are available on the R, I, B, G, and C fonts — "fi", fl", and "ff". They may be input by "\(fi", "\(fl", and "\(ff" respectively, or an automatic ligature mode may be invoked.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.lg N	on	on	no	Ligature mode is turned on, if N is absent or non-zero; and turned off, if N = 0. When ligature mode is on the character pairs "fi", "fl", and "ff" are converted to the single characters "fi", "fl", and "ff" respectively at input time. Such conversion does not occur during the reading of requests, macro and string names, macro and string definitions, file names, and number register names.

IV. Special characters. Non-ASCII characters on both the standard fonts and the special mathematical font are input by means of the sequence "\(xx" where "\(" is the character name indicator and "xx" is a two-character character name; these characters are shown together with thier names in Table II.

V. Backspacing, underlining, overstriking, etc. Underlining and generalized line-drawing are discussed in §XI.*IV*. A generalized overstriking function is described in the same section. The ASCII backspace character is replaced (except in copy mode) by a backward horizontal motion having the width of the most recent non-backspace character.

VI. Control characters (normally "." and "'"). Both control characters may be changed. In addition, it is possible to specify that a certain number of input lines are to be taken literally as text (non-control) lines. Another way to "begin" a text line with a control character is to precede that character with the non-printing, zero-width filler character input as "\&". Still another way is to specify output translation of some convenient character into the control character (see "tr" request later).

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.cc c	.	.	no	The basic control character is set to "c" or reset to ".". Use of this request to temporarily change the control character can result in requests in line-trap-invoked macros being misinterpreted.
.c2 c	'	'	no	The nobreak control character is set to "c" or reset to "'". See warning under "cc".
.li N	-	N=1	no	Accept the next N input lines *at the current string/macro input level (or higher levels)* as literal text.

VII. Number arithmetic. A simple form of arithmetic expression can be used anywhere that a number is expected while processing a request. The operators permitted are + (addition), − (subtraction), * (multiplication), / (division), and unary minus. Evaluation is from left to right and no grouping is permitted. For example, if number register x contains −4, the "number" 7*\nx+2/13 evaluates to −2. It should be remembered that in certain contexts an initial + or − is taken as an incremental designator, and that *each* number in an expression should have suitable unit indicators attached (see *Explanation* section).

VIII. Output translation. Provision exists for specifying a mapping of any character into any other character. All text processing (e. g. character comparisons) takes place with the original 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*	*Cause Break*	*Explanation*
.tr abcd....	none	-	no	Translate. "a" will be mapped into "b", "c" will be mapped into "d", etc. If an odd number of characters is given, the last one will be mapped into the space character.

A common use of the "tr" request is to provide nonadjustable spaces. When normal filling and adjusting is done the space character between words indicates where the additional space may be put and where line splitting may occur. A ".tr |", which specifies that "|" be mapped into a space during output, permits tying two or more words together so that they will neither be moved apart nor split across two lines. Examples might be "Fig.|12", "Mr.|Smith", and "2|+|x|=|y". It should be noted that horizontal tabs which are converted to pure horizontal motion (represented internally by a character) and the narrow space character \| both provide a nonadjustable horizontal distance.

IX. Transparent Throughput. An input line beginning with a "\!" is transparently output (except for the initial "\!"); only "copy mode" processing takes place (see §VI.*II*). This mechanism is used to pass control information to a post-processor or to pass TROFF requests to a macro during a diversion.

X. Local Horizontal and Vertical Motions, and the Width Function

I. Local Motions. The escape sequences \u, \d, and \r may be used to obtain local vertical motions 1/2 Em up, 1/2 Em down, and 1 Em up respectively. More generally \v'N' and \h'N' generate local vertical and horizontal motions respectively. The number N can be negative, can be generated by a number register, can be generated by the width function (see below), and can involve number arithmetic. The positive vertical direction is downward. For example, "E^{2}" could be generated by the sequence "E\s8\v'-.5m'2\v'.5m'\s10"; it should be noted in this example that the 0.5 Em vertical motions are at the 8-point size.

II. Width Function. A width function of the form \w'string' may be used to generate a number equal to the size (in basic units) of "string". For example, ".ti −\w'1. 'u" could be used to temporarily indent leftward the exact distance equal to the size of the string "1. " (note the necessary scale indicator "u"). Another example can be seen in the word "Version" in the page header of this document in which the characters "Ve" are moved closer together (taking advantage of the shape of "V") by using "V\h'−\w'V'u/5u'e" to generate "Ve" in which the "e" is moved closer to the "V" by 1/5th of the width of the latter.

The width function also sets the number register "ct" 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.

III. Mark horizontal place. A \ka appearing within an input line will result in the then current horizontal position (in the input line) being stored in number register "a". For example, the sequence "\kaword\h'|\nau'" will end by returning to the beginning of "word".

XI. Overstrike, Bracket, Line-drawing, and Zero-width Functions

I. Overstriking. Automatic overstriking of two to nine characters may be accomplished using the overstrike function which has the form \o'string'. The string may not include motion characters (e. g. \h'N', \v'N', etc.). The characters in the string are overprinted in a centered fashion with a width equal to the width of the widest character in the string.

II. Zero-width characters. The automatic spacing of printed characters may be suppressed for a single character by use of the Zero-Width function which has the form \zc where "c" is the character affected.

III. Large Brackets. The Special Mathematical Font contains a number of pieces of large brackets (⎧ ⎨ ⎩ ⎪ ⎡ ⎢ ⎣ ⎤ ⎥ ⎦) that can be combined into various bracket styles. A function having the form \b'string' may be used to pile up vertically the characters in "string" (the first character on top and the last at the bottom); the characters are vertically separated by 1 Em and the total pile is centered 1/2 Em above the current baseline.

IV. Line drawing. The function \l'Nc' will draw a string of "c"s for a distance N. (\(lower case L) *not* \(one)). If "c" is not specified, the "_" (rule) is used. 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 left end of the string. In the case of connectible characters (_ (rule), _ (underrule), and ‾ (root en)) the remainder space is covered by over-lapping. As an example, a macro to underline a string can be written

```
.de ul
\\$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

```
.ul "underlined words"
```

and

```
.bx "words in a box"
```

yields underlined words and words in a box.

XII. Hyphenation.

The automatic hyphenation may be turned both off and on. When on it may be suppressed for a single word. In addition, the permissible hyphenation points within a word can be specified by imbedding a hyphenation indicator character within a word.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.nh	on	-	no	No hyphenation. Automatic hyphenation is turned off. Words containing hyphens (e. g. mother-in-law) may still be split across lines.

.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.

XIII. Three Part Titles.

A titling function provides for automatic placing of three fields respectively at the left, center, and right of a specified title line length. Use of "tl" has no effect on current line accumulation.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.tl 'left'center'right'		-	no	Title. The strings represented by "left", "center", and "right" are respectively left adjusted, centered, and right adjusted within the current title length. Any of the fields may be empty. If the character "%" is found within any of the fields it is replaced by the current page number in the current number style (see requests "ar", "ro", and "RO"). Any graphic character may be used in place of the field delimiter "'".
.lt N	6.5 Inches	prev	no	Length of title. The length of lines and titles are maintained separately. Indents do not apply to titles; page-offsets do.

"tl" is usually used within header and footer macros. For example, ".tl ''– % –'" will print the page number in the center of the title length.

XIV. Output Line Numbering.

Automatic sequence numbering of output lines may be turned both on and off. When on, a line number is prepended to output lines. Blank lines and other vertical spaces are not numbered. The
3 prepended entity has the general form: (I units of number indent) plus (a three digit number with leading zeros printed as units of space) plus (S separating units of space). The prepended entity effectively offsets the line which still has the current line length; a reduction in line length is neces-
6 sary, if the right margin is to be preserved. The unit of space is the width of a number digit (typically 1 En). In addition, 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 fields). The parameters I, S,
9 and M are controllable.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.nm ±N M S I		off	no	Number mode. If ±N is given, line numbering is turned on; the first line numbered is numbered ±N. If any of the remaining parameters are given, they will be set to the given values; an alphabetic character causes the corresponding parameter to be unaffected. Default values are M=1, S=1, and I=0. In the absence of all arguments, numbering is turned off; the next line number is preserved for possible further use.
.np M S I	1,1,1	reset	no	Number parameters are set or reset to default values (see above) in the absence of all arguments. Individually absent parameters (or specified by an alphabetic character) are unchanged.

As an example, the paragraph portions 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
12 front of this paragraph; and ".nm" finally placed at the end. Line lengths were also changed (by 4n) to keep the right side aligned. Some other examples are: ".nm +5 5 x 3" turns on numbering with the line number of the next line to be 5 greater than the last numbered line, with M=5, with spacing

15 S untouched, and with the indent I set to 3; ".np 3" sets M=3 and leaves S and I alone.

XV. Conditional Input Line Acceptance

Input lines may be accepted conditionally.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.if c anything		-	no	See next.
.if !c anything		-	no	"
.if N anything		-	no	"
.if !N anything		-	no	If. "anything" is an arbitrary input line; it can be either text or a request. "c" is a one-character, built-in condition name. "N" is any number; it can be an expression involving number registers. If the condition is "true", or if the number is greater than zero, the remainder of the line containing "anything" is accepted as input, otherwise the rest of the line is ignored. Any spaces in front of "anything" are ignored. If "c" or "N" are prefaced by "!" (not), the line is accepted when the condition is false or the number is less than or equal to zero.

Built-in Conditions.

Name	*Meaning*
o	The current page number is odd.
e	The current page number is even.
t	The formatter is TROFF.
n	The formatter is NROFF.

Some examples are:

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

outputs a title if the page number is even; and

```
.if !\na-\nb .xy
```

invokes the macro "xy" if the number (\na–\nb) is zero or negative.

XVI. 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.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.ev N	N=0	prev	no	Environment is switched to environment N. Switching is done in push-down fashion (limit of 5) so that restoring a previous environment should be done with ".ev" rather than specific reference. There are 3 environments; N can be 0, 1, or 2.

The different environments all have the same initial default parameter values. Parameters within an environment are those associated with:

character point size	filling	number register indicator
character font	title length	tab replacement character
vertical line space	centering count	leader replacement character
extra line space	line numbering	partially collected words
line length	tab settings	partially collected lines
indenting	hyphenation control	
adjusting	request control characters	

Everything else is global — i. e. not switched when environments are. Examples of global entities include the page offset, page numbers, current line number, number registers, line trap tables, and macro definitions. It may be noted that partially collected words and lines are kept with an environment so that environment switching prevents the next break from printing the previous environment's partial line.

XVII. Insertions from the Standard Input Stream

The input stream in TROFF can be switched to the system standard input stream, which normally is the user's keyboard, but which may be a pipe or a file. The input stream is switched back to its original source when two newline characters in a row are encountered (i. e. an "extra" blank line is found). This mechanism is useful for form-letter-like documentation. With UNIX's ability to switch the Standard Input to a file, insertions can be stored in a file.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.rd prompt	-	no prompt	no	Read insert. TROFF input is switched to UNiX standard Input until two newline characters in a row are found. The extra newline is thrown away. If "prompt", a character string without blanks, is given, "prompt" is written out on the user's typewriter to indicate that the input is requested. If "prompt" is absent, a Bell character is written instead. Because "rd" behaves like a macro, arguments may be placed after "prompt".
.ex	-	-	no	Exit. Text processing is finished exactly as if all input was finished.

The contents of "prompt" should be chosen to suggest what or which insertion is currently wanted. Prompting is automatically suppressed when the standard input is not a console typewriter. The input text and inserts via "rd" *should not* simultaneously come from the standard input. Multiple copies of a processed "letter" are easily obtained by causing "letter" to reinvoke itself by means of the "nx" request (see below) and including an "ex" request in an insert after the end of the last letter.

XVIII. Input File Switching

At any given instant, input is taken from either the current input file (the top input level) or from a macro or string (at some macro/string invocation level). The following requests permit input file switching at the top level.

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.so filename		-	no	Switch Source file. Input at the top (input file) level is switched to the file named "filename". The current input level is not changed. When the new file ends, input is again taken from the original file beginning with the line after the "so" request.
.nx filename		-	no	Next file is "filename". No further input is taken from the current input file (or current input level); "filename" becomes the current input file and the input level is popped back to the top (input file) level. This request may be used to re-

peatedly process the same file by having the nextfile be itself.

XIX. Miscellaneous

Request Form	*Initial Value*	*If no Argument*	*Cause Break*	*Explanation*
.tm string	-	-	no	Typewriter message. "string" is printed on the user's console typewriter, if any (output is *not* buffered). "string" is read in copy mode (see §VI.*II*).
.hs N	on	on	no	Control high-speed multiple scan mode (N=0 for off). TROFF normally prints every output line a number of times equal to the number of different point-sizes that occur in the line. This strategy significantly speeds up the printing because changing sizes is *slower* than the extra motion involved. This multiple scanning may be turned off if it is desired to improve the registration between characters having different sizes.
.ig	-	-	no	Ignore. Input lines are ignored up to and including the next input line beginning with "..". The ignored lines are read in copy mode and some side effects can occur (e. g. \n+a will increment register "a").
.fl	-	-	yes	Flush output buffer. This request is useful for debugging request sequences because it can be used to force output that typically would be buffered and hidden.
.ab	-	-	no	Abort. This request causes an IOT trap and causes a core image to be produced. It is used for debugging the TROFF program itself.

XX. Error Messages.

Various error messages may be written on the user's typewriter. If the latter cannot be found, the message is written on the shell's output file. If that cannot be found, it is written in a file called "tr.out". This strategy is also used by "tm" for writing its message, and by "rd" for writing out the prompt. If an error is considered fatal, TROFF will attempt to exit gracefully. Examples of non-fatal messages are "Word overflow" and "Line overflow" which indicate that some word didn't fit in the word buffer (in filled text) and that a line being generated became too long for the line buffer respectively; in both cases what didn't fit was discarded.

TUTORIAL EXAMPLES

Introduction

The following examples will range from the provision of simple headers and footers, to the provision of more general ones, to programming multi-column output, and finally to programming for footnotes. The term programming is used here because using TROFF is more like building upon a framework of basic features than like choosing from a list of specific features. For example, TROFF has no built-in footnote mechanism, but such a mechanism can be "programmed" using the basic macro, diversion, environment switching, and line-trap mechanisms. Nonprogrammers and others who may not want to probe TROFF possibilities should find it relatively easy to use TROFF for simple formatting jobs such as documents requiring straightforward header and footer designs. To use footnotes it is only necessary to include at the beginning of a document an available pre-canned set of macro definitions that implement a footnote mechanism. Similar pre-canned macros are available for multi-column output.

1. Headers and Footers

The material that fills the space between the top of a page and the beginning of the running text is termed a "header" and is typically relatively constant from one page to the next. The material that fills the space between the bottom of the running text and the bottom of a page is termed the "footer". It is necessary to define two macros — one each describing the header and footer respectively. Then it is necessary to specify at what line on the page they are to be invoked. For example

```
.de hd
'sp 1i
.ns
..
.de fo
'bp
..
.wh 0 hd
.wh -1i fo
```

describes a header macro named "hd" which produces 1 inch of space (without causing a break), and describes a footer macro named "fo" that will simply eject the page (without causing a break). The "wh" requests specify that "hd" is to be invoked at the beginning of the page (making "hd" a header) and that "fo" is to be invoked 1 inch *from the bottom*. The break suppression (using the "'") prevents what is left over from the last line printed on a page from being printed by the occurrence of a break. The ".ns" turns on the no-space mode that suppresses vertical space that would result from the accidental occurrence of an ".sp N" exactly at the location of the intended first output text line on a page.

The trap at the top of the first page occurs at the first encountered break function or when regular text is first encountered; the definition of any header must occur before this if it is to appear on the first page.

The sizes of these headers and footers could be parameterized by the following alternative definitions

```
.nr t 1i
.nr b -1i
.de hd
'sp \\ntu
.ns
..
.de fo
'bp
..
.wh 0 hd
.wh \nbu fo
```

which achieves the same end except that the margin sizes are initialized in two number registers. Note that the number register references have the necessary "u" (units) scale factor to prevent scaling by the default factor (VS in these cases). Using "\\" in "'sp \\nt" results in the stored definition of "hd" contain-

ing "'sp \nt" causing the top margin to depend each time on the then current value of number register "t"; this permits easy dynamic modification of the top margin by changing the register "t" (using the "nr" request). Dynamic modification of the bottom margin requires the use of the "ch" request prior to the desired change; for example, ".ch fo \nbu−3" pushes the trap for "fo" up 3 line spacings (since the default scaling for the "3" is the VS). The footer can be arranged to reset itself by including in the definition for "fo" a ".ch fo \\nbu".

With the above kind of footer something like

```
.de pp
.tl ''- % -''
..
.wh −0.5i pp
```

would place an independently positioned, centered page number a half-inch from the bottom.

A more typical header macro might look like

```
.de hd
'sp 0.5i
.ps10
.ftR
.tl ''- % -''
.ps
.ft
'sp  1i
.ns
..
```

and produce a centered page number at the top of the page. Here the setting (and restoring) of the point size and font is done to insure that the page number is in the desired size and font in spite of what is happening in the main body of the text. In cases where the footer may print something, the saving of the size and font would be there. Or like

```
.de hd
.ps10
.ftR
.tl '\(rn'''
'sp |0.5i−2
.if e .tl 'Page %'''
.if o .tl '''Page %'
.tl ''Some Title''
.ps
.ft
'sp |1i
.ns
..
```

which produces a cut mark (root en) at the top of the page, a left adjusted page number on even numbered pages, a right adjusted number on odd pages, and centers a title on the next line. The first "sp" supplies a space equal to a half-inch from the top minus 2 VSs to cause the two title lines to end a half-inch from the top. The second "sp" moves down to 1 inch from the top.

Footer macros can of course contain more than a "'bp" although it is convenient for them to end that way to facilitate changing trap locations.

```
.de fo
'sp 0.25i
.tl ''Bottom Title''
'bp
..
```

is a simple example.

The above examples all involved headers or footers which avoided producing a break. The more general case is exemplified by

```
.de hd
.ev 1
(Any kind of text and
 text processing)
.ev
..
```

which switches to another environment to avoid any conflict with the main stream of text processing.

II. Major Headings and Paragraphs

It is most convenient to define macros to be placed at the beginning of paragraphs and various headings. For example,

```
.de pg
.ftR
.sp 0.4
.ne 2
.ti 0.3i
..
.de mh
.sp 0.5
.ne 3.4
.ft B
..
```

defines a paragraph macro named "pg" which forces the "R" (Times Roman) font in case it has been left something else, spaces 0.4 VSs, requires that 2 VSs of space be left on the page, and sets up a temporary indent of 0.3 inches. And defines a major heading macro "mh" that spaces 0.5 VSs, requires that 3.4 VSs of space be left on the page (room for a one line heading, a 0.4 VS space, and two lines of the following paragraph), and set the font to "B" (Times Bold).

The use of such macros is an example of the good practice of including macros rather than explicit TROFF requests in the main body of text. This provides the maximum ease of changing the global format of a document.

III. Labeled Indented Paragraphs

Another common formatting problem is properly placing labels on indented paragraphs when the label must go in the indent space on the same line as the first line of the paragraph. Assumming that the paragraph is filled text it is necessary that the white space around (and possibly within) the label must not contain space characters that can be expanded in size for adjustment purposes. Among many solutions is:

```
.in N2
.ta N1 N2
(tab)label(tab)words in paragraph...
...
.in 0
```

N1 is the distance to the label and N2 is the indent as well as a tab stop for filling in the space between the label and the first word of the paragraph. Any spaces within the label would usually be characters translated into space using the "tr" request. Fancier label arrangments are possible using the field mechanism (see §VIII.*11*).

IV. Multiple Column Output

It is relatively easy to arrange for multiple column output. The following shows a header and footer set of macros containing the additional requests for generating 3 column output.

```
.de fo
.if !\\n+c-3 .nc
.if \\nc-3 .np
..
.de nc
.po +2.25i
.rt
..
.de np
.po 26i/27u
'sp .25i
.ps10
.ftR
.if !\\n%-1 .tl "- % -"
'bp
..
.de hd
.tl '\(rn''
'sp |0.75i-1
.if \\n%-1 .tl "- % -"
.ft
.ps
'sp |1.0i
.ns
.nr c 1 1
.mk
..
.wh 0 hd
.wh -1.0i fo
.ll 2.0i
.br
```

The header macro, "hd", sets a number register, "c", to an initial column number (1) and specifies that any autoincrementing be by 1. The request "mk" marks the place at the bottom of the header as the return point for a subsequent "rt" request. The footer, "fo", increments the column number and tests it to see whether a new column or new page is to be next and invokes either the macro "nc" or "np" accordingly. The new column macro "nc" increments the page offset by 2.25 inches and returns to the marked place. The new page macro "np" resets the page offset to the original value of 26/27 inches and goes on to perform normal footer functions. The line length is set to 2 inches. The three columns are each 2 inches wide and are spaced 0.25 inches apart for a total page width of 6.5 inches. The only change necessary to get a different number of columns would be to change the line length, incremental page offset, and the constant against which the column number is compared in the footer macro.

These macros also show how one can number page one at the bottom of the page and subsequent pages at the top.

V. Generating Footnotes

The programming example to be discussed here implements a fairly general footnote mechanism. One aim is to define a set of macros that permits simple user demarcation of footnote content. A user of the footnote macros to be described needs only to include the following

```
.fn
(Any kind of text and
```

text processing)
.ef

as close after the point of footnote reference as possible. The macro "fn" indicates the beginning of the footnote, and "ef" indicates end of footnote. The footnote text is processed in another environment while being diverted for later use.

A usable footnote program is:

```
.nr bm 1i
.de hd HEADER
.tl '\(rn'''
'sp |0.5i
.tl 'Head title'''
.ft
.ps
'sp |1i
.nr x 0 1
.nr y 0-\\n(bm
.if \\n(dn .fz
.ns
..
.de fo FOOTER
.nr dn 0
.if \\nx .xf
.ch fo -\\n(bm
.ftR
.ps10
'bp
..
.de bo
.tl '- % -'
..
.de fn BEGIN FOOTNOTE
.da FN
.ev1
.if !\\n+x-1 .fs
..
.de ef END FOOTNOTE
.br
.ev
.di
.if !\\nx-1 .nr dn +\\n(.v
.nr y -\\n(dn
.ch fo \\nyu
.if \\n(nl+\\n(.v-\\n(.p-\\ny .ch fo \\n(nlu+\\n(.vu
..
.de fs SEPARATOR
\l'1i'
.br
..
.de fz
.fn
.nf
.fy
.fi
```

```
.ef
..
.de fx
.di fy
..
.de xf
.ev1
.nf
.FN
.rm FN
.di
.ev
..
.wh 0 hd
.wh -\\n(bm fo
.wh -0.5i bo
.ch fo 12i
.wh -\\n(bm fx
.ch fo -\\n(bm
.ev1
'll5.5i
.ps8
'vs10p
.ev
.br
```

The size of the bottom margin is specified in number register "bm". The header "hd" initializes two registers, "x" and "y", at the top of every page; "x" is the basic per-page footnote counter and "y" is used during footnote output to keep track of where the footer macro should be sprung. The conditional invocation of macro "fz" reprocesses the remainder of any footnote that did not fit at the bottom of the previous page. The footer tests whether or not any footnotes were processed and if so, invokes "xf". Afterwards the position trap for "fo" is reset to place "bm". The macro "xf" switches environments, sets the "nofill" mode, interpolates in the footnotes (macro "FN"), removes "FN", terminates any possible diversion of footnote material that did not fit on the page, restores fill mode, and restores the previous environment. The macro "fx", also planted at position "bm", will save the portion of the last footnote that didn't fit, if any, by diverting into a macro "fy". The latter is preprocessed in the header, if there has been any diversion (non-zero number register "dn") since the beginning of the footer. The begin-footnote macro "fn" diverts the footnote (in append fashion) into the macro "FN" and switches environments. If the footnote to be processed is the first one on a page, the footnote separator macro "fs" is invoked. In this example, "fs" merely generates a 1 inch rule. The end-footnote macro "ef" resets the environment, ends the diversion, and pushes up the trap position for "fo" to account for the size of the footnote. The trap is moved up the size of the footnote plus, if the footnote is the first one on the page, the size of the current line spacing. If this movement would move the trap up past the current vertical place plus the current line spacing, the trap is moved up only to the latter place; this occurrence ordinarily will result in the last footnote being split between two pages. The "wh" and "ch" requests plant the header trap at the top of the page, plant the footer trap at position "bm", move the footer trap somewhere past the page length, plant a trap for "fx" also at position "bm", and finally move the footer trap back. The two macros "fo" and "fx" are effectively planted at the same place; the trap for "fx" can occur only if the footer trap is moved up by the occurrence of a footnote, because it is further down the internal trap list; it was necessary to temporarily move the trap for "fo" to avoid "fx" replacing "fo" at that trap position.

Table I

Font Style Examples

Times Roman

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
! # $ & ( ) ' ' * + − . , / : ; = ? @ [ ] |
● □ — - - _ ¼ ½ ¾ fi fl ff – ¶ ° † § ′

Times Italic

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
*! # $ & ( ) ' ' * + − . , / : ; = ? @ [ ] |*
● □ — - - _ ¼ ½ ¾ *fi fl ff* – ¶ ° † § ′

Times Bold

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
! # $ & ( ) ' ' * + − . , / : ; = ? @ [ ] |
● □ — - - _ ¼ ½ ¾ **fi fl ff** – ¶ ° † § ′

Special Mathematical Font

" ′ \ ^ _ ` ~ / < > { }
α β γ δ ϵ ζ η θ ι κ λ μ ν ξ ο π ρ σ ς τ υ ϕ χ ψ ω
Γ Δ Θ Λ Ξ Π Σ Υ Φ Ψ Ω
√ ⩾ ⩽ ≡ ≈ ≃ ∼ ≠ → ← ↑ ↓ ∴ × ÷ ± ∪ ∩ ⊂ ⊃ ⊆ ⊇ ∞ ∂ ∇
¬ ∫ ∝ ∅ ∈ ∉ ® © | ¢ ‡ ☛ ☚ ☎ 🕭 ∣ ○ ⌈ ⌉ ⌊ ⌋ { } ⌊ ⌋ ⌈ ⌉

In the above, non-alphanumeric characters are separated by ¼ Em space.

Table II

Escape (Naming) Conventions for ´, `, —, and _ and for Non-ASCII Special Characters

Non-ASCII characters and "—" on the regular fonts.

Char	*Escape*	*Name*
‘	`	open quote
’	'	close quote
—	\(em	¼ Em dash
-	-	hyphen
—	\(em	¼ Em dash
-	\(hy	hyphen
•	\(bu	bullet
□	\(sq	square
_	\(ru	rule
¼	\(14	1/4
½	\(12	1/2

Char	*Escape*	*Name*
¾	\(34	3/4
−	\(mi	minus
fi	\(fi	fi
fl	\(fl	fl
ff	\(ff	ff
–	\(en	en dash
¶	\(pg	paragraph
°	\(de	degree
†	\(dg	dagger
§	\(sc	section
’	\(fm	foot mark

Non-ASCII characters and ´, `, and _ on the Special font.

The ASCII characters ", ´, `, <, >, \, {, }, ~, ^, and _ exist only on the Special Font and are printed as a 1-Em space if that Font is not mounted. The following characters exist only on the special font except for the upper case Greek letter names followed by † which are mapped into upper case English letters in the Times Roman font.

Char	*Escape*	*Name*
´	\(aa	acute accent
`	\(ga	grave accent
_	\(ul	underrule
/	\(sl	slash (matching backslash)
α	\(*a	alpha
β	\(*b	beta
γ	\(*g	gamma
δ	\(*d	delta
ε	\(*e	epsilon
ζ	\(*z	zeta
η	\(*y	eta
θ	\(*h	theta
ι	\(*i	iota
κ	\(*k	kappa
λ	\(*l	lambda
μ	\(*m	mu
ν	\(*n	nu
ξ	\(*c	xi
ο	\(*o	omicron
π	\(*p	pi
ρ	\(*r	rho

Char	*Escape*	*Name*
σ	\(*s	sigma
ς	\(ts	terminal sigma
τ	\(*t	tau
υ	\(*u	upsilon
φ	\(*f	phi
χ	\(*x	chi
ψ	\(*q	psi
ω	\(*w	omega
A	\(*A	Alpha†
B	\(*B	Beta†
Γ	\(*G	Gamma
Δ	\(*D	Delta
E	\(*E	Epsilon†
Z	\(*Z	Zeta†
H	\(*Y	Eta†
Θ	\(*H	Theta
I	\(*I	Iota†
K	\(*K	Kappa†
Λ	\(*L	Lambda
M	\(*M	Mu†
N	\(*N	Nu†

Char	*Escape*	*Name*
Ξ	\(*C	Xi
O	\(*O	Omicron†
Π	\(*P	Pi
P	\(*R	Rho†
Σ	\(*S	Sigma
T	\(*T	Tau†
Υ	\(*U	Upsilon
Φ	\(*F	Phi
X	\(*X	Chi†
Ψ	\(*Q	Psi
Ω	\(*W	Omega
√	\(sr	square root
_	\(rn	root en extender
≥	\(>=	>=
≤	\(<=	<=
≡	\(==	identically equal
≈	\(~~	approx equal
≃	\(~=	approx =
~	\(ap	approximates
≠	\(!=	not equal
→	\(->	right arrow
←	\(<-	left arrow
↑	\(ua	up arrow
↓	\(da	down arrow
∴	\(tf	therefore
×	\(mu	multiply
÷	\(di	divide
±	\(+-	plus-minus
∪	\(cu	cup (union)
∩	\(ca	cap (intersection)
⊂	\(sb	subset of
⊃	\(sp	superset of
⊆	\(ib	improper subset
⊇	\(ip	improper superset
∞	\(if	infinity
∂	\(pd	partial derivative
∇	\(gr	gradient
¬	\(no	not
∫	\(is	integral sign
∝	\(pt	proportional to
∅	\(es	empty set
∈	\(mo	member of
∉	\(nm	not a member
®	\(rg	registered
©	\(co	copyright
|	\(br	box vertical rule
¢	\(ct	cent sign
‡	\(dd	double dagger
☛	\(rh	right hand
☚	\(lh	left hand
☎	\(tp	telephone
Ⓐ	\(bs	bell system sign

Char	*Escape*	*Name*
|	\(or	or
○	\(ci	circle
⎧	\(lt	left top of big curly bracket
⎩	\(lb	left bottom
⎫	\(rt	right top
⎭	\(rb	right bot
⎨	\(lk	left center of big curly bracket
⎬	\(rk	right center of big curly bracket
⎪	\(bv	bold vertical
⌊	\(lf	left floor (left bottom of big square bracket)
⌋	\(rf	right floor (right bottom)
⌈	\(lc	left ceiling (left top)
⌉	\(rc	right ceiling (right top)