NROFF/TROFF User's Manual

Joseph F. Ossanna
(updated for 4.3BSD by Mark Seiden)

Bell Laboratories
Murray Hill, New Jersey 07974

Introduction

NROFF and TROFF are text processors under the UNIX Time-Sharing System that format text for typewriter-like terminals and for a Graphic Systems phototypesetter, respectively. (Device-independent TROFF, part of the Documenter's Workbench, supports additional output devices.) They accept lines of text interspersed with lines of format control information and format the text into a printable, paginated document having a user-designed style. NROFF and TROFF offer 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.

NROFF and TROFF 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. NROFF can prepare output directly for a variety of terminal types and is capable of utilizing the full resolution of each terminal.

Usage

The general form of invoking NROFF (or TROFF) at UNIX command level is

nroff *options files* (or **troff** *options files*)

where *options* represents any of a number of option arguments and *files* 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 files, are:

Option	*Effect*
−**i**	Read standard input after the input files are exhausted.
−**m***name*	Prepends the macro file **/usr/lib/tmac.***name* to the input *files*.
−**n***N*	Number first generated page *N*.
−**o***list*	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;* a initial −*N* means from the beginning to page *N;* and a final *N*– means from *N* to the end.
−**q**	Invoke the simultaneous input-output mode of the **rd** request.
−**r***aN*	Number register *a* (one-character) is set to *N*.
−**s***N*	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 newline. TROFF will stop the phototypesetter every *N* pages, produce a trailer to allow changing cassettes, and will resume after the phototypesetter START button is pressed.
−**z**	Efficiently suppress formatted output. Only produce output to standard error (from **tm** requests or diagnostics).

NROFF Only

−T*name* Specifies the name of the output terminal type. Currently defined names are **37** for the (default) Model 37 Teletype®, **tn300** for the GE TermiNet 300 (or any terminal without half-line capabilities), **300S** for the DASI-300S, **300** for the DASI-300, and **450** for the DASI-450 (Diablo Hyterm).

−e Produce equally-spaced words in adjusted lines, using full terminal resolution.

−h On output, use tabs during horizontal spacing to increase speed. Device tabs setting are assumed to be (and input tabs are initially set to) every 8 character widths.

TROFF Only

−a Send a printable (ASCII) approximation of the results to the standard output.

−b TROFF will report whether the phototypesetter is busy or available. No text processing is done.

−f Refrain from feeding out paper and stopping phototypesetter at the end of the run.

−t Direct output to the standard output instead of the phototypesetter.

−w Wait until phototypesetter is available, if currently busy.

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

nroff −o*4,8−10* **−T***300S* **−m***abc file1 file2*

requests formatting of pages 4, 8, 9, and 10 of a document contained in the files named *file1* and *file2*, specifies the output terminal as a DASI-300S, and invokes the macro package *abc*.

Various pre- and post-processors are available for use with NROFF and TROFF. These include the equation preprocessors NEQN and EQN[1] (for NROFF and TROFF respectively), and the table-construction preprocessor TBL[2]. A reverse-line postprocessor COL[3] is available for multiple-column NROFF output on terminals without reverse-line ability; COL expects the Model 37 Teletype escape sequences that NROFF produces by default. TK[3] is a 37 Teletype simulator postprocessor for printing NROFF output on a Tektronix 4014. TC[5] is a phototypesetter-simulator postprocessor for TROFF that produces an approximation of phototypesetter output on a Tektronix 4014. For example, in

tbl *files* **| eqn | troff −t** *options* **| tc**

the first **|** indicates the piping of TBL's output to EQN's input; the second the piping of EQN's output to TROFF's input; and the third indicates the piping of TROFF's output to TC.

The remainder of this manual consists of: a Summary and outline; a Reference Manual keyed to the outline; and a set of Tutorial Examples. Another tutorial is [5].

References

[1] B. W. Kernighan, L. L. Cherry, *Typesetting Mathematics — User's Guide (Second Edition),* Bell Laboratories.

[2] M. E. Lesk, *Tbl — A Program to Format Tables,* Bell Laboratories internal memorandum.

[3] Internal on-line documentation (*man* pages) on UNIX.

[4] B. W. Kernighan, *A TROFF Tutorial*, Bell Laboratories.

[5] Your site may have similar programs for more modern displays.

SUMMARY OF REQUESTS AND OUTLINE OF THIS MANUAL

Request Form	*Initial Value**	*If No Argument*	*Notes#*	*Explanation*

1. General Explanation

2. Font and Character Size Control

Request Form	*Initial Value**	*If No Argument*	*Notes#*	*Explanation*
.ps ±*N*	10 point	previous	E	Point size; also **\s**±*N*.†
.fz *F* ±*N*	off	-	E	font *F* to point size ±*N*.
.fz S *F* ±*N*	off	-	E	Special Font characters to point size ±*N*.
.ss *N*	12/36 em	ignored	E	Space-character size set to *N*/36 em.†
.cs *F N M*	off	-	P	Constant character space (width) mode (font *F*).†
.bd *F N*	off	-	P	Embolden font *F* by *N*−1 units.†
.bd S *F N*	off	-	P	Embolden Special Font when current font is *F*.†
.ft *F*	Roman	previous	E	Change to font *F* = *x*, *xx*, or 1-4. Also **\f***x*, **\f(***xx*, **\f***N*.
.fp *N F*	R,I,B,S	ignored	-	Font named *F* mounted on physical position 1≤*N*≤4.

3. Page Control

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

4. Text Filling, Adjusting, and Centering

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

5. Vertical Spacing

Request Form	*Initial Value**	*If No Argument*	*Notes#*	*Explanation*
.vs *N*	1/6in;12pts	previous	E,**p**	Vertical base line spacing (*V*).
.ls *N*	*N*=1	previous	E	Output *N*−1 *V*s after each text output line.
.sp *N*	-	*N*=1*V*	B,**v**	Space vertical distance *N in either direction*.
.sv *N*	-	*N*=1*V*	**v**	Save vertical distance *N*.
.os	-	-	-	Output saved vertical distance.
.ns	space	-	D	Turn no-space mode on.
.rs	-	-	D	Restore spacing; turn no-space mode off.

6. Line Length and Indenting

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

7. 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 a macro.

*Values separated by "**;**" are for NROFF and TROFF respectively.

#Notes are explained at the end of this Summary and Index

†No effect in NROFF.

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

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
.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	D	Divert output to macro *xx*.
.da *xx*	-	end	D	Divert and append to *xx*.
.wh *N xx*	-	-	**v**	Set location trap; negative is w.r.t. page bottom.
.ch *xx N*	-	-	**v**	Change trap location.
.dt *N xx*	-	off	D,**v**	Set a diversion trap.
.it *N xx*	-	off	E	Set an input-line count trap.
.em *xx*	none	none	-	End macro is *xx*.

8. Number Registers

.nr *R* ±*N M*	-	-	**u**	Define and set number register *R*; auto-increment by *M*.
.af *R c*	arabic	-	-	Assign format to register *R* (*c*=**1**, **i**, **I**, **a**, **A**).
.rr *R*	-	-	-	Remove register *R*.

9. Tabs, Leaders, and Fields

.ta *Nt* ...	0.8; 0.5in	none	E,**m**	Tab settings; *left* type, unless *t*=**R**(right), **C**(centered).
.tc *c*	none	none	E	Tab repetition character.
.lc *c*	**.**	none	E	Leader repetition character.
.fc *a b*	off	off	-	Set field delimiter *a* and pad character *b*.

10. Input and Output Conventions and Character Translations

.ec *c*	\	\	-	Set escape character.
.eo	on	-	-	Turn off escape character mechanism.
.lg *N*	-; on	on	-	Ligature mode on if *N*>0.
.ul *N*	off	*N*=1	E	Underline (italicize in TROFF) *N* input lines.
.cu *N*	off	*N*=1	E	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*	**.**	**.**	E	Set control character to *c*.
.c2 *c*	**´**	**´**	E	Set nobreak control character to *c*.
.tr *abcd....*	none	-	O	Translate *a* to *b*, etc. on output.

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

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

13. Hyphenation.

.nh	hyphenate	-	E	No hyphenation.
.hy *N*	hyphenate	hyphenate	E	Hyphenate; *N* = mode.
.hc *c*	**\%**	**\%**	E	Hyphenation indicator character *c*.
.hw *word1 ...*		ignored	-	Exception words.

14. Three Part Titles.

.tl *´left´center´right´*		-	-	Three part title.
.pc *c*	**%**	off	-	Page number character.
.lt ±*N*	6.5 in	previous	E,**m**	Length of title.

15. Output Line Numbering.

.nm ±*N M S I*	off	E	Number mode on or off, set parameters.	
.nn *N*	-	*N*=1	E	Do not number next *N* lines.

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
16. Conditional Acceptance of Input				
.if *c anything*	-	-		If condition *c* true, accept *anything* as input, for multi-line use \\{*anything* \\}.
.if !*c anything*	-	-		If condition *c* false, accept *anything*.
.if *N anything*	-	**u**		If expression *N* > 0, accept *anything*.
.if !*N anything*		-	**u**	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*	-	**u**		If portion of if-else; all above forms (like **if**).
.el *anything*		-	-	Else portion of if-else.
17. Environment Switching.				
.ev *N*	*N*=0	previous	-	Environment switched (*push down*).
18. Insertions from the Standard Input				
.rd *prompt*	-	*prompt*=BEL		Read insertion.
.ex	-	-	-	Exit from NROFF/TROFF.
19. Input/Output File Switching				
.so *filename*		-	-	Switch source file *(push down)*.
.nx *filename*		end-of-file	-	Next file.
.pi *program*		-	-	Pipe output to *program* (NROFF only).
20. Miscellaneous				
.mc *c N*	-	off	E,**m**	Set margin character *c* and separation *N*.
.tm *string*	-	newline	-	Print *string* on terminal (UNIX standard error output).
.ig *yy*	-	*.yy*=**..**	-	Ignore till call of *yy*.
.pm *t*	-	all	-	Print macro names and sizes; if *t* present, print only total of sizes.
.ab *string*	-	-	-	Print a message and abort.
.fl	-	-	B	Flush output buffer.
21. Output and Error Messages				

Notes-

B	Request normally causes a break.
D	Mode or relevant parameters associated with current diversion level.
E	Relevant parameters are a part of the current environment.
O	Must stay in effect until logical output.
P	Mode must be still or again in effect at the time of physical output.
v,**p**,**m**,**u**	Default scale indicator; if not specified, scale indicators are *ignored*.

Alphabetical Request and Section Number Cross Reference

ab 20	c2 10	di 7	ex 18	hw 13	lg 10	ne 3	os 5	rd 18	ss 2	uf 10
ad 4	cc 10	ds 7	fc 9	hy 13	li 10	nf 4	pc 14	rm 7	sv 5	ul 10
af 8	ce 4	dt 7	fi 4	ie 16	ll 6	nh 13	pi 19	rn 7	ta 9	vs 5
am 7	ch 7	ec 10	fl 20	if 16	ls 5	nm 15	pl 3	rr 8	tc 9	wh 7
as 7	cs 2	el 16	fp 2	ig 20	lt 14	nn 15	pm 20	rs 5	ti 6	
bd 2	cu 10	em 7	ft 2	in 6	mc 20	nr 8	pn 3	rt 3	tl 14	
bp 3	da 7	eo 10	fz 2	it 7	mk 3	ns 5	po 3	so 19	tm 20	
br 4	de 7	ev 17	hc 13	lc 9	na 4	nx 19	ps 2	sp 5	tr 10	

Escape Sequences for Characters, Indicators, and Functions

<table>
<tr><th>Section Reference</th><th>Escape Sequence</th><th>Meaning</th></tr>
<tr><td>10.1</td><td>\\</td><td>\ (to prevent or delay the interpretation of \ )</td></tr>
<tr><td>10.1</td><td>\e</td><td>Printable version of the current escape character.</td></tr>
<tr><td>2.1</td><td>\´</td><td>´ (acute accent); equivalent to \(aa</td></tr>
<tr><td>2.1</td><td>\`</td><td>` (grave accent); equivalent to \(ga</td></tr>
<tr><td>2.1</td><td>\−</td><td>− Minus sign in the current font</td></tr>
<tr><td>7</td><td>\.</td><td>Period (dot) (see de)</td></tr>
<tr><td>11.1</td><td>\(space)</td><td>Unpaddable space-size space character</td></tr>
<tr><td>11.1</td><td>\0</td><td>Digit width space</td></tr>
<tr><td>11.1</td><td>\|</td><td>1/6 em narrow space character (zero width in NROFF)</td></tr>
<tr><td>11.1</td><td>\^</td><td>1/12 em half-narrow space character (zero width in NROFF)</td></tr>
<tr><td>4.1</td><td>\&</td><td>Non-printing, zero width character</td></tr>
<tr><td>10.6</td><td>\!</td><td>Transparent line indicator</td></tr>
<tr><td>10.7</td><td>\"</td><td>Beginning of comment</td></tr>
<tr><td>7.3</td><td>\$N</td><td>Interpolate argument 1≤N≤9</td></tr>
<tr><td>13</td><td>\%</td><td>Default optional hyphenation character</td></tr>
<tr><td>2.1</td><td>\(xx</td><td>Character named xx</td></tr>
<tr><td>7.1</td><td>*x, *(xx</td><td>Interpolate string x or xx</td></tr>
<tr><td>9.1</td><td>\a</td><td>Non-interpreted leader character</td></tr>
<tr><td>12.3</td><td>\b´abc...´</td><td>Bracket building function</td></tr>
<tr><td>4.2</td><td>\c</td><td>Interrupt text processing</td></tr>
<tr><td>11.1</td><td>\d</td><td>Forward (down) 1/2 em vertical motion (1/2 line in NROFF)</td></tr>
<tr><td>2.2</td><td>\fx,\f(xx,\fN</td><td>Change to font named x or xx, or position N</td></tr>
<tr><td>11.1</td><td>\h´N´</td><td>Local horizontal motion; move right N (negative left)</td></tr>
<tr><td>11.3</td><td>\kx</td><td>Mark horizontal input place in register x</td></tr>
<tr><td>12.4</td><td>\l´Nc´</td><td>Horizontal line drawing function (optionally with c)</td></tr>
<tr><td>12.4</td><td>\L´Nc´</td><td>Vertical line drawing function (optionally with c)</td></tr>
<tr><td>8</td><td>\nx,\n(xx</td><td>Interpolate number register x or xx</td></tr>
<tr><td>12.1</td><td>\o´abc...´</td><td>Overstrike characters a, b, c, ...</td></tr>
<tr><td>4.1</td><td>\p</td><td>Break and spread output line</td></tr>
<tr><td>11.1</td><td>\r</td><td>Reverse 1 em vertical motion (reverse line in NROFF)</td></tr>
<tr><td>2.3</td><td>\sN, \s±N</td><td>Point-size change function</td></tr>
<tr><td>9.1</td><td>\t</td><td>Non-interpreted horizontal tab</td></tr>
<tr><td>11.1</td><td>\u</td><td>Reverse (up) 1/2 em vertical motion (1/2 line in NROFF)</td></tr>
<tr><td>11.1</td><td>\v´N´</td><td>Local vertical motion; move down N (negative up)</td></tr>
<tr><td>11.2</td><td>\w´string´</td><td>Interpolate width of string</td></tr>
<tr><td>5.2</td><td>\x´N´</td><td>Extra line-space function (negative before, positive after)</td></tr>
<tr><td>12.2</td><td>\zc</td><td>Print c with zero width (without spacing)</td></tr>
<tr><td>16</td><td>\{</td><td>Begin conditional input</td></tr>
<tr><td>16</td><td>\}</td><td>End conditional input</td></tr>
<tr><td>10.7</td><td>\(newline)</td><td>Concealed (ignored) newline</td></tr>
<tr><td>-</td><td>\X</td><td>X, any character not listed above</td></tr>
</table>

The escape sequences \\, **\.**, \", **\$**, *, **\a**, **\n**, **\t**, and \(newline) are interpreted in *copy mode* (§7.2).

Predefined General Number Registers

Section Reference	*Register Name*	*Description*
3	**%**	Current page number.
19	**c.**	Number of *lines* read from current input file.
11.2	**ct**	Character type (set by *width* function).
7.4	**dl**	Width (maximum) of last completed diversion.
7.4	**dn**	Height (vertical size) of last completed diversion.
-	**dw**	Current day of the week (1-7).
-	**dy**	Current day of the month (1-31).
11.3	**hp**	Current horizontal place on *input* line (not in ditroff)
15	**ln**	Output line number.
-	**mo**	Current month (1-12).
4.1	**nl**	Vertical position of last printed text base-line.
11.2	**sb**	Depth of string below base line (generated by *width* function).
11.2	**st**	Height of string above base line (generated by *width* function).
-	**yr**	Last two digits of current year.

Predefined Read-Only Number Registers

Section Reference	*Register Name*	*Description*
7.3	**.$**	Number of arguments available at the current macro level.
-	**.A**	Set to 1 in TROFF, if −**a** option used; always 1 in NROFF.
11.1	**.H**	Available horizontal resolution in basic units.
5.3	**.L**	Set to current *line-spacing* (**ls**) parameter
-	**.P**	Set to 1 if the current page is being printed; otherwise 0.
-	**.T**	Set to 1 in NROFF, if −**T** option used; always 0 in TROFF.
11.1	**.V**	Available vertical resolution in basic units.
5.2	**.a**	Post-line extra line-space most recently utilized using **\x**´*N*´.
19	**.c**	Number of *lines* read from current input file.
7.4	**.d**	Current vertical place in current diversion; equal to **nl**, if no diversion.
2.2	**.f**	Current font as physical quadrant (1-4).
4	**.h**	Text base-line high-water mark on current page or diversion.
6	**.i**	Current indent.
4.2	**.j**	Current adjustment mode and type.
4.1	**.k**	Length of text portion on current partial output line.
6	**.l**	Current line length.
4	**.n**	Length of text portion on previous output line.
3	**.o**	Current page offset.
3	**.p**	Current page length.
2.3	**.s**	Current point size.
7.5	**.t**	Distance to the next trap.
4.1	**.u**	Equal to 1 in fill mode and 0 in nofill mode.
5.1	**.v**	Current vertical line spacing.
11.2	**.w**	Width of previous character.
-	**.x**	Reserved version-dependent register.
-	**.y**	Reserved version-dependent register.
7.4	**.z**	Name of current diversion.

REFERENCE MANUAL

1. General Explanation

1.1. 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 **.** (period) or ´ (acute accent)—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 control character ´ suppresses the *break* function—the forced output of a partially filled line—caused by certain requests. The control character may be separated from the request/macro name by white space (spaces and/or tabs) for æsthetic reasons. Names must be followed by either space or newline. Control lines with unrecognized names are ignored.

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

1.2. Formatter and device resolution. TROFF internally uses 432 units/inch, (for historical reasons, corresponding to the Graphic Systems phototypesetter which had a horizontal resolution of 1/432 inch and a vertical resolution of 1/144 inch.) NROFF internally uses 240 units/inch, corresponding to the least common multiple of the horizontal and vertical resolutions of various typewriter-like output devices. TROFF rounds horizontal/vertical numerical parameter input to its own internal horizontal/vertical resolution. NROFF similarly rounds numerical input to the actual resolution of the output device indicated by the **−T** option (default Model 37 Teletype).

1.3. Numerical parameter input. Both NROFF and TROFF accept numerical input with the scale indicator suffixes 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	$6\times S$	C
n	En = Em/2	$3\times S$	*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		

In NROFF, *both* the em and the en are taken to be equal to the C, which is output-device dependent; common values are 1/10 and 1/12 inch. 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 **|** may be prefixed 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* (§7.4) to 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.

1.4. Numerical expressions. Wherever numerical input is expected, an expression involving parentheses, the arithmetic operators **+**, **−**, **/**, **∗**, **%** (mod), and the logical operators **<**, **>**, **<=**, **>=**, **=** (or **==**), **&** (and), **:** (or) may be 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.

1.5. 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.

2. Font and Character Size Control

2.1. Character set. The TROFF character set consists of a typesetter-dependent basic character set plus a Special Mathematical Font character set—each having 102 characters. An example of these character sets is shown in the Appendix Table I. All printable ASCII characters are included, with some on the Special Font. With three exceptions, these ASCII characters are input as themselves, and non-ASCII characters are input in the form **\(***xx* where *xx* is a two-character name given in the Appendix Table II. The three ASCII exceptions are mapped as follows:

ASCII Input		Printed by TROFF	
Character	Name	Character	Name
´	acute accent	’	close quote
`	grave accent	‘	open quote
−	minus	-	hyphen

The characters ´, `, and − may be input by \´, \`, and \− respectively or by their names (Table II). The ASCII characters **@**, **#**, **"**, **´**, **`**, **<**, **>**, ****, **{**, **}**, **˜**, **ˆ**, and **_** exist only on the Special Font and are printed as a 1-em space if that font is not mounted.

NROFF understands the entire TROFF character set, but can in general print only ASCII characters, additional characters as may be available on the output device, such characters as may be able to be constructed by overstriking or other combination, and those that can reasonably be mapped into other printable characters. The exact behavior is determined by a driving table prepared for each device. The characters ´, `, and _ print as themselves.

2.2. Fonts. The default mounted fonts are Times Roman (**R**), Times Italic (**I**), Times Bold (**B**), and the Special Mathematical Font (**S**) on physical typesetter positions 1, 2, 3, and 4 respectively. These fonts are used in this document. The *current* font, initially Roman, may be changed (among the mounted fonts) by use of the **ft** request, or by imbedding at any desired point either **\f***x*, **\f(***xx*, or **\f***N* where *x* and *xx* are the name of a mounted font and *N* is a numerical font position. It is *not* necessary to change to the Special Font; characters on that font are automatically handled. A request for a named but not-mounted font is *ignored*. TROFF can be informed that any particular font is mounted by use of the **fp** request. The list of known fonts is installation dependent. In the subsequent discussion of font-related requests, *F* represents either a one/two-character font name or the numerical font position, 1-4. The current font is available (as numerical position) in the read-only number register **.f**.

NROFF understands font control and normally underlines Italic characters (see §10.5).

2.3. Character size. Character point sizes available are typesetter dependent, but often include 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. The **ps** request is used to change or restore the point size. Alternatively the point size may be changed between any two characters by imbedding a **\s***N* at the desired point to set the size to N, or a **\s**$\pm N$ ($1 \le N \le 9$) to increment/decrement the size by N; **\s0** restores the *previous* size. Requested point size values that are between two valid sizes yield the larger of the two. The current size is available in the **.s** register. NROFF ignores type size control.

Request Form	*Initial Value*	*If No Argument*	*Notes**	*Explanation*
.ps $\pm N$	10 point	previous	E	Point size set to $\pm N$. Alternatively imbed **\s**N or **\s**$\pm N$. Any positive size value may be requested; if invalid, the next larger valid size will result, with a maximum of 36. A paired sequence $+N, -N$ will work because the previous requested value is also remembered. Ignored in NROFF.
.fz $F \pm N$	off	-	E	The characters in font F will be adjusted to be in size $\pm N$. Characters in the Special Font encountered during the use of font F will have the same size modification. (Use the **.fz S** request if different treatment of Special Font characters is required). **.fz** must follow any **.fp** request for the position.
.fz S $F \pm N$	off	-	E	The characters in the Special Font will be in size $\pm N$ independent of previous **.fz** requests.
.ss N	12/36 em	ignored	E	Space-character size is set to N/36 ems. This size is the minimum word spacing in adjusted text. Ignored in NROFF.
.cs $F\,N\,M$	off	-	P	Constant character space (width) mode is set on for font F (if mounted); the width of every character will be 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 an 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. The mode must be still or again in effect when the characters are physically printed. Ignored in NROFF.
.bd $F\ N$	off	-	P	The characters in font F will be artificially emboldened by printing each one twice, separated by $N-1$ basic units. A reasonable value for N is 3 when the character size is in the vicinity of 10 points. If N is missing the embolden mode is turned off. The column heads above were printed with **.bd I 3**. The mode must be still or again in effect when the characters are physically printed. Ignored in NROFF.
.bd S $F\ N$	off	-	P	The characters in the Special Font will be emboldened whenever the current font is F. This manual was printed with **.bd S B 3**. The mode must be still or again in effect when the characters are physically printed.
.ft F	Roman	previous	E	Font changed to F. Alternatively, imbed **\f**F. The font name **P** is reserved to mean the previous font.
.fp $N\ F$	R,I,B,S	ignored	-	Font position. This is a statement that a font named F is mounted on position N (1-4). It is a fatal error if F is not known. 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. The default mounting sequence assumed by TROFF is R, I, B, and S on positions 1, 2, 3 and 4.

*Notes are explained at the end of the Summary and Index above.

3. 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). See §7 and Tutorial Examples §T2. 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* (§7.4) 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 the Graphic Systems phototypesetter was about 7.54 inches, beginning about 1/27 inch from the left edge of the 8 inch wide, continuous roll paper, but these characteristics are typesetter- dependent. The physical limitations on NROFF output are output-device dependent.

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.pl ±*N*	11 in	11 in	**v**	Page length set to ±*N*. The internal limitation is about 75 inches in TROFF and about 136 inches in NROFF. The current page length is available in the **.p** register.
.bp ±*N*	*N*=1	-	B*,**v**	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; 26/27 in†	previous	**v**	Page offset. The current *left margin* is set to ±*N*. The TROFF initial value provides about 1 inch of paper margin including the physical typesetter margin of 1/27 inch. In TROFF the maximum (line-length)+(page-offset) is about 7.54 inches. See §6. The current page offset is available in the **.o** register.
.ne *N*	-	*N*=1 *V*	D,**v**	Need *N* vertical space. If the distance, *D*, to the next trap position (see §7.5) 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	D	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	D,**v**	Return *upward only* to a marked vertical place in the current diversion. If ±*N* (w.r.t. current place) 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 absolute place stored in a explicit register; e. g. using the sequence **.mk** *R* ... **.sp** **|\n***R***u**.

4. Text Filling, Adjusting, and Centering

4.1. Filling and adjusting. Normally, words are collected from input text lines and assembled into a output text line until some word doesn't fit. An attempt is then made to hyphenate the word to assemble a part of it into the output

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

†Values separated by ";" are for NROFF and TROFF respectively.

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 (§2). 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 (§13) 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 text base-line high-water mark (lowest place) on the current page is in the **.h** register. The **.k** register (read-only) contains the horizontal size of the text portion (without indent) of the current partially-collected output line (if any) in the current environment.

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 (§10.4) can be made to not look like a control line by preceding it by the non-printing, zero-width filler character **\&**. Still another way is to specify output translation of some convenient character into the control character using **tr** (§10.5).

4.2. Interrupted text. The copying of a 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	-	-	B	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	-	B,E	Fill subsequent output lines. The register **.u** is 1 in fill mode and 0 in nofill mode.
.nf	fill on	-	B,E	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	E	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. The type indicator can also be a value saved from the read-only **.j** number register, which is set to contain the current adjustment mode and type.

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	-	E	Noadjust. 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	B,E	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.

5. Vertical Spacing

5.1. 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; TROFF default is 10-point type on a 12-point spacing (as in this document). The current *V* is available in the **.v** register. Multiple-*V* line separation (e. g. double spacing) may be requested with **ls**.

5.2. 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.

5.3. 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 may be reserved using **sv**.

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.vs *N*	1/6in;12pts	previous	E,**p**	Set vertical base-line spacing size *V*. Transient *extra* vertical space available with **\x**´*N* ´ (see above).
.ls *N*	*N*=1	previous	E	*Line* spacing set to ±*N*. *N*−1 *V*s *(blank lines)* are appended to each output text line. The (read-only) number register **.L** is set to contain the current line-spacing value. Appended blank lines are omitted, if the text or previous appended blank line reached a trap position.
.sp *N*	-	*N*=1*V*	B,**v**	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*=1*V*	**v**	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 (see **os**). 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	-	D	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	-	D	Restore spacing. The no-space mode is turned off.
Blank text line.		-	B	Causes a break and outputs a blank line just like **sp 1**.

6. Line Length and Indenting

The maximum line length for fill mode may be set with **ll**. The indent may be 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 line length and indent are available in registers **.l** and **.i** respectively. The length of *three-part titles* produced by **tl** (see §14) is *independently* set by **lt**.

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.ll ±*N*	6.5 in	previous	E,**m**	Line length is set to ±*N*. In TROFF the maximum (line-length)+(page-offset) is about 7.54 inches.
.in ±*N*	*N*=0	previous	B,E,**m**	Indent is set to ±*N*. The indent is prepended to each output line.
.ti ±*N*	-	ignored	B,E,**m**	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.

7. Macros, Strings, Diversion, and Position Traps

7.1. 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 newline 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 strings *x* and *xx* are interpolated at any desired point with *****x* and ***(***xx* respectively. String references and macro invocations may be nested.

7.2. 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 newlines indicated by \(newline) are eliminated.
- Comments indicated by \" are eliminated.
- **\t** and **\a** are interpreted as ASCII horizontal tab and SOH respectively (§9).
- \\ is interpreted as \.
- **\.** is interpreted as "**.**".

These interpretations can be suppressed by prepending a \. 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.

7.3. 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 newline 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 *N*th argument (1≤*N*≤9). If an invoked argument doesn't exist, a null string results. For example, the macro *xx* may be defined by

```
.de xx      \"begin definition
Today is \\$1 the \\$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 \. 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 push down, 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 \ ) to delay interpolation until argument reference time.

7.4. Diversions. Processed output may be diverted into a macro for purposes such as footnote processing (see Tutorial §T5) 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 with the *transparent* mechanism described in §10.6.

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).

7.5. Traps. Three types of trap mechanisms are available—page traps, a diversion trap, and an input-line-count trap. Macro-invocation traps may be planted using **wh** at 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 (see Tutorial Examples §T5). 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 the **it** request 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	D	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	D	Divert, appending to *xx* (append version of **di**).
.wh *N xx*	-	-	**v**	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*	-	-	**v**	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	D,**v**	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	E	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 inline 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.

8. Number Registers

A variety of parameters are available to the user as predefined, named *number registers* (see Summary and Index, page 7). 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* (§1.4).

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
\n–x	x decremented 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*	-	-	**u**	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 (example 2 above). The read-only registers and the *width* function (§11.2) 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.

9. Tabs, Leaders, and Fields

9.1. 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*.

9.2. 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 **#** and the padding indicator is **^**, **#^***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	E,**m**	Set tab stops and types. *t*=**R**, right adjusting; *t*=**C**, centering; *t* absent, left adjusting. TROFF tab stops are preset every 0.5in.; NROFF every 8 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	E	The tab repetition character becomes *c*, or is removed specifying motion.
.lc *c*	**.**	none	E	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.

10. Input and Output Conventions and Character Translations

10.1. Input character translations. Ways of inputting the graphic character set were discussed in §2.1. The ASCII control characters horizontal tab (§9.1), SOH (§9.1), and backspace (§10.3) are discussed elsewhere. The newline 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** (§10.5). *All* others are ignored.

The *escape* character \ introduces *escape sequences*—causes the following character to mean another character, or to indicate some function. A complete list of such sequences is given in the Summary and Index on page 6. \ 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 be used 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.

10.2. Ligatures. Five ligatures are available in the current TROFF character set — **fi**, **fl**, , , and . They may be input (even in NROFF) by **\(fi**, **\(fl**, **\(ff**, **\(Fi**, and **\(Fl** respectively. The ligature mode is normally on in TROFF, and *automatically* invokes ligatures during input.

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.lg *N*	off; on	on	-	Ligature mode is turned on if *N* is absent or non-zero, and turned off if *N*=0. If *N*=2, only the two-character ligatures are automatically invoked. Ligature mode is inhibited for request, macro, string, register, or file names, and in *copy mode*. No effect in NROFF.

10.3. 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. Underlining as a form of line-drawing is discussed in §12.4. A generalized overstriking function is described in §12.1.

NROFF automatically underlines characters in the *underline* font, specifiable with **uf**, normally Times Italic on font position 2 (see §2.2). In addition to **ft** and **\f***F*, 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 (italicize in TROFF) 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** (§14) *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	E	A variant of **ul** that causes *every* character to be underlined in NROFF. Identical to **ul** in TROFF.
.uf *F*	Italic	Italic	-	Underline font set to *F*. In NROFF, *F* may *not* be on position 1 (initially Times Roman).

10.4. Control characters. Both the control character **.** and the *no-break* control character **´** 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*	**.**	**.**	E	The basic control character is set to *c*, or reset to "**.**".
.c2 *c*	´	´	E	The *nobreak* control character is set to *c*, or reset to "´".

10.5. 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	-	O	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.

10.6. 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-processor or to imbed control lines in a macro created by a diversion.

10.7. Comments and concealed newlines. 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 \. The sequence \(newline) is *always* ignored—except in a comment. Comments may be imbedded at the *end* of any line by prefacing them with **"**. The newline 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 **.\"**.

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

11.1. Local Motions. The functions **\v**´*N*´ and **\h**´*N*´ can be used for *local* vertical and horizontal motion respectively. The distance *N* may be negative; the *positive* directions are *rightward* and *downward*. A *local* motion is one contained *within* a line. To avoid unexpected vertical dislocations, it is necessary that the *net* vertical local motion within a word in filled text and otherwise within a line balance to zero. The above and certain other escape sequences providing local motion are summarized in the following table.

<table>
<tr><th rowspan="2">Vertical
Local Motion</th><th colspan="2">Effect in</th><th rowspan="2">Horizontal
Local Motion</th><th colspan="2">Effect in</th></tr>
<tr><th>TROFF</th><th>NROFF</th><th>TROFF</th><th>NROFF</th></tr>
<tr><td>\v´N´</td><td colspan="2">Move distance N</td><td>\h´N´
\(space)
\0</td><td colspan="2">Move distance N
Unpaddable space-size space
Digit-size space</td></tr>
<tr><td>\u
\d
\r</td><td>½ em up
½ em down
1 em up</td><td>½ line up
½ line down
1 line up</td><td>\|
\^</td><td>1/6 em space
1/12 em space</td><td>ignored
ignored</td></tr>
</table>

As an example, **E**$^{\mathbf{2}}$ could be generated by the sequence **E\s−2\v´−0.4m´2\v´0.4m´\s+2**; it should be noted in this example that the 0.4 em vertical motions are at the smaller size.

11.2. 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 TROFF 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.

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

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

12.1. 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. *string* should *not* contain local vertical motion. As examples, **\o´e\´´** produces **é**, and **\o´\(mo\(sl´** produces ∉.

12.2. Zero-width characters. The function **\z***c* will output *c* without spacing over it, and can be used to produce left-aligned overstruck combinations. As examples, **\z\(ci\(pl** will produce ⊕ and **\(br\z\(rn\(ul\(br** will produce the smallest possible constructed box □.

12.3. Large Brackets. The Special Mathematical Font contains a number of bracket construction pieces (⎧ ⎩ ⎫ ⎭ ⎨ ⎬ ⎪ ⌊ ⌋ ⌈ ⌉) that can be combined into various bracket styles. The function **\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 (½ line in NROFF). For example, **\b´\(lc\(lf´E\|\b´\(rc\(rf´\x´−0.5m´\x´0.5m´** produces $\left[\mathbf{E}\right]$.

12.4. Line drawing. The function **\l**´*Nc*´ 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"

yield underlined words and words in a box.

The function **\L**´*Nc*´ will draw a vertical line consisting of the (optional) character *c* stacked vertically apart 1 em (1 line in NROFF), 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 any 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* compensating motions are made; the instantaneous 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 possibly 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.

13. 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), em-dashes (**\(em**), or hyphenation indicator characters—such as mother-in-law—are *always* subject to splitting after those characters, whether or not automatic hyphenation is on or off.

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.nh	hyphenate	-	E	Automatic hyphenation is turned off.
.hy*N*	on,N=1	on,N=1	E	Automatic hyphenation is turned on for $N \geq 1$, 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 values are additive; i. e. $N=14$ will invoke all three restrictions.
.hc *c*	**\%**	**\%**	E	Hyphenation indicator character is set to *c* or to the default **\%**. The indicator does not appear in the output.
.hw *word1 ...*		ignored	-	Specify hyphenation points in words with imbedded minus signs. Versions 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.

14. Three Part Titles.

The titling function **tl** provides for automatic placement of three fields at the left, center, and right of a line with a title-length specifiable with **lt**. **tl** may be used anywhere, and is independent of the normal text collecting process. A common use is in header and footer macros.

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.tl *´left´center´right´*		-	-	The strings *left*, *center*, and *right* are respectively left-adjusted, centered, and right-adjusted in the current title-length. Any of the strings may be empty, and overlapping is permitted. If the page-number character (initially **%**) is found within any of the fields it is replaced by the current page number having the format assigned to register **%**. Any character may be used as the string delimiter.
.pc *c*	**%**	off	-	The page number character is set to *c*, or removed. The page-number register remains **%**.
.lt ±*N*	6.5 in	previous	E,**m**	Length of title set to ±*N*. The line-length and the title-length are *independent*. Indents do not apply to titles; page-offsets do.

15. Output Line Numbering.

Automatic sequence numbering of output lines may be requested with **nm**. When in effect, a three-digit, arabic number plus a 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 the right margin aligned with an earlier margin. Blank lines, other vertical spaces, and lines generated by **tl** are *not* numbered. Numbering can be temporarily suspended with **nn**, 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 fields).

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.nm ±*N M S I*	off		E	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 further use in number register **ln**.
.nn *N*	-	*N*=1	E	The next *N* text output lines are not numbered.

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 front of this paragraph; and **.nm** finally placed at the end. Line lengths were also changed (by **\w´0000´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 numbered line, with *M*= 5, with spacing *S* untouched, and with the indent *I* set to 3.

16. 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*	-	**u**		If expression *N* > 0, accept *anything*.
.if !*N anything*		-	**u**	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*	-	**u**		If portion of if-else; all above forms (like **if**).
.el *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.

17. 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 E 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≤*N*≤2. Switching is done in push-down fashion so that restoring a previous environment *must* be done with **.ev** rather than specific reference.

18. Insertions from the Standard Input

The input can be temporarily switched to the system *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 UNIX, 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 like a macro, and arguments may be placed after *prompt*.
.ex	-	-	-	Exit from NROFF/TROFF. 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** (§19); the process would ultimately be ended by an **ex** in the insertion file.

19. Input/Output File Switching

The (read-only) number register **.c** contains the input line number in the current input file. The number register **c.** is a general register serving the same purpose.

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 occurs immediately. 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*.

20. Miscellaneous

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.mc *c N*	-	off	E,**m**	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 nofill mode) the character will be appended to the line. If *N* is not given, the previous *N* is used; the initial *N* is 0.2 inches in NROFF and 1 em in TROFF. 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. (see §21).
.ig *yy*	-	*.yy*=**..**	-	Ignore input lines. **ig** behaves exactly like **de** (§7) 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.
.ab *string*	-	-	-	Print *string* on standard error and terminate immediately. The default *string* is "User Abort". Does not cause a break. Only output preceding the last break is written.
.fl	-	-	B	Flush output buffer. Used in interactive debugging to force output.

21. Output and Error Messages.

The output from **tm**, **pm**, **ab** and the prompt from **rd**, as well as various *error* messages are written onto UNIX's *standard error* output. The latter is different from the *standard output*, where NROFF formatted output goes. By 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 and TROFF. 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 a $*$ in NROFF and a $\Leftarrow$ in TROFF. 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

T1. Introduction

Although NROFF and TROFF have 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 as page margins and footnotes are deliberately not built into NROFF and TROFF. 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. In many cases, number registers would really be used to reduce the number of places where numerical information is kept, and to concentrate conditional parameter initialization like that which depends on whether TROFF or NROFF is being used.

T2. Page Margins

As discussed in §3, *header* and *footer* macros are usually defined to describe the top and bottom page margin 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 0 hd
.wh −1i 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 (§3). 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 ´ 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 \{\
´sp |0.5i−1     \"tl base at 0.5i
.tl ´´− % −´´   \"centered page number
.ps             \"restore size
.ft             \"restore font
.vs  \}         \"restore vs
´sp |1.0i       \"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, font, 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. If TROFF is used, a *cut mark* is drawn in the form of *root-en*'s at each margin. 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

*For example: P. A. Crisman, Ed., *The Compatible Time-Sharing System,* MIT Press, 1965, Section AH9.01 (Description of RUNOFF program on MIT's CTSS system).

and restore both the current *and* previous values as shown for size in the following:

```
.de fo
.nr s1 \\n(.s       \"current size
.ps
.nr s2 \\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
```

T3. 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 requests a temporary indent.

```
.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 1 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 output device vertical resolution, 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 S 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** (§8).

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
..
```

The intended usage is "**.lp** *label*"; *label* will begin at 0.2 inch, and cannot exceed a length of 0.3 inch 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.

T4. 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 fo              \"footer
.ie \\n+(cl<2 \{\
.po +3.4i           \"next column; 3.1+0.3
.rt                 \"back to mark
.ns \}              \"no-space mode
.el \{\
.po \\nMu           \"restore left margin
```

```
.  ---
´bp \}
..
.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.

T5. 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
.if \\n(dn .fz      \"leftover footnote
..
.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
.if "\\n(.z"fy" .di \"end overflow diversion
.nr x 0             \"disable fx
.ev  \}             \"pop 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+\\ny) \
.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 0 hd            \"header trap
.wh 12i fo          \"footer trap, temp position
.wh –\\nbu fx       \"fx at footer position
.ch fo –\\nbu       \"conceal fx 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.

T6. The Last Page

After the last input file has ended, NROFF and TROFF invoke the *end macro* (§7), 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 I

Font Style Examples

The following fonts are printed in 12-point, with a vertical spacing of 14-point, and with non-alphanumeric characters separated by ¼ em space (all measurements on 8.5 × 11 inch paper prior to photoreduction). This font sample is printed on an APS-5 phototypesetter at University of California, Berkeley.

Times Roman

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
! $ % & ( ) ‘ ’ * + − . , / : ; = ? [ ] |
● □ — - _ ¼ ½ ¾ fi fl ° † ′ ¢ ® ©

Times Italic

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
*! $ % & ( ) ‘ ’ * + − . , / : ; = ? [ ] |*
● □ — - _ ¼ ½ ¾ fi fl ° † ′ ¢ ® ©

Times Bold

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
! $ % & ( ) ‘ ’ * + − . , / : ; = ? [ ] |
● □ — - _ ¼ ½ ¾ fi fl ° † ′ ¢ ® ©

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
√ ‾ ≥ ≤ ≡ ∼ ≈ ≠ → ← ↑ ↓ × ÷ ± ∪ ∩ ⊂ ⊃ ⊆ ⊇ ∞ ∂
§ ∇ ¬ ∫ ∝ ∅ ∈ ‡ ⇒ ⇐ ♥ | ○ ⎛ ⎝ ⎞ ⎠ ⎨ ⎬ ⎪ ⌊ ⌋ ⌈ ⌉ │

Table II

Input Naming Conventions for ´, `, and −
and for Non-ASCII Special Characters

Non-ASCII characters and *minus* on the standard fonts.

Char	*Input Name*	*Character Name*	Char	*Input Name*	*Character Name*
’	´	close quote	ﬁ	\(fi	ﬁ
‘	`	open quote	ﬂ	\(fl	ﬂ
—	\(em	3/4 Em dash		\(ff	ff
-	−	hyphen or		\(Fi	ffi
-	\(hy	hyphen		\(Fl	ffl
−	\−	current font minus	°	\(de	degree
•	\(bu	bullet	†	\(dg	dagger
□	\(sq	square	′	\(fm	foot mark
_	\(ru	rule	¢	\(ct	cent sign
¼	\(14	1/4	®	\(rg	registered
½	\(12	1/2	©	\(co	copyright
¾	\(34	3/4			

Non-ASCII characters and ´, `, _, +, −, =, and ∗ on the special font.

The ASCII characters @, #, ", ´, `, <, >, \, {, }, ˜, ˆ, 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 whatever font is mounted on font position one (default Times Roman). The special math plus, minus, and equals are provided to insulate the appearance of equations from the choice of standard fonts.

Char	*Input Name*	*Character Name*	Char	*Input Name*	*Character Name*
+	\(pl	math plus	λ	\(*l	lambda
−	\(mi	math minus	μ	\(*m	mu
=	\(eq	math equals	ν	\(*n	nu
∗	\(**	math star	ξ	\(*c	xi
§	\(sc	section	ο	\(*o	omicron
´	\(aa	acute accent	π	\(*p	pi
`	\(ga	grave accent	ρ	\(*r	rho
_	\(ul	underrule	σ	\(*s	sigma
/	\(sl	slash (matching backslash)	ς	\(ts	terminal sigma
α	\(*a	alpha	τ	\(*t	tau
β	\(*b	beta	υ	\(*u	upsilon
γ	\(*g	gamma	φ	\(*f	phi
δ	\(*d	delta	χ	\(*x	chi
ε	\(*e	epsilon	ψ	\(*q	psi
ζ	\(*z	zeta	ω	\(*w	omega
η	\(*y	eta	Α	\(*A	Alpha†
θ	\(*h	theta	Β	\(*B	Beta†
ι	\(*i	iota	Γ	\(*G	Gamma
κ	\(*k	kappa	Δ	\(*D	Delta

Char	*Input Name*	*Character Name*
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†
Σ	\(*S	Sigma
Τ	\(*T	Tau†
Υ	\(*U	Upsilon
Φ	\(*F	Phi
Χ	\(*X	Chi†
Ψ	\(*Q	Psi
Ω	\(*W	Omega
√	\(sr	square root
‾	\(rn	root en extender
≥	\(>=	>=
≤	\(<=	<=
≡	\(==	identically equal
≈	\(~=	approx =
∼	\(ap	approximates
≠	\(!=	not equal
→	\(->	right arrow
←	\(<-	left arrow
↑	\(ua	up arrow
↓	\(da	down arrow
×	\(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
│	\(br	box vertical rule
‡	\(dd	double dagger
⇒	\(rh	right hand
⇐	\(lh	left hand
♥	\(bs	Bell System logo (typesetter-dependent)
│	\(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)

Char	Input Name	Character Name	Char	Input Name	Character Name