Heirloom Documentation Tools
Nroff/Troff User's Manual

Joseph F. Ossanna
Brian W. Kernighan
Gunnar Ritter
and others

Introduction

nroff and *troff* are text processors under the UNIX Time-Sharing System[1] that format text for typewriter-like terminals and for a typesetter/raster devices, respectively. 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.

troff produces its output in a device-independent form, although parameterized for a specific device; *troff* output must be processed by a driver for that device to produce printed output.

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.

On the Heirloom Documentation Tools Edition

In Summer 2005, Sun Microsystems, Inc. released the source code to the Solaris system,[6] including the System V Release 4 version of *troff,* a derivative of AT&T *Documenter's Workbench troff,* version 2. It had undergone few changes since the end of the 1980's, so it could serve as a clean starting point for a new version of *troff* which is intended to be highly compatible with UNIX *troff,* but which also provides additional features desirable for a high-quality typesetting application at the beginning of the 21st century.

As with the other components of the *Heirloom Project,* the original code, once it had been released under an Open Source license, has been made portable such that it compiles and runs on the contemporary UNIX-style systems, including Linux. It continues to be freely available under the same license as originally released, including its complete source code.

PostScript and its close relative PDF are now the only device languages which are relevant to high-quality printing; actually, PostScript itself is more and more becoming an intermediate language for the generation of PDF documents. The *Heirloom* version of *troff* is thus primarily directed towards generating PostScript output for further processing by a PDF creator, such as Ghostscript or Adobe Distiller; it can generate PDF-specific instructions for prepress usage as well as for online navigation in PDF documents.

The principal output device independence of *troff* has nevertheless been retained, and changes to the intermediate language have been minor. Many *troff* post-processors will thus continue to be usable with no or little adaptions.

PostScript Type 1, OpenType, and TrueType have become device-independent font formats; virtually all commercial and free fonts are available in one of them. There is thus no need for a *troff*-specific device-independent font format anymore; instead, *Heirloom troff* can read font metrics directly from Type 1, OpenType, and TrueType font files. This has greatly relieved the task of installing fonts—it suffices to copy the original files to a user-selectable font directory—, and makes it possible to access advanced typographic data, such as kerning tables or substitution instructions for old-style numerals.

troff provides convenient access to any character in a font file either by its PostScript name, by its Unicode position as specified in a font-specific or a generic table, or by conversion from POSIX-style locale-specific characters to Unicode positions. The last form allows direct input of international language texts in almost any character encoding, including UTF-8.

Improvements to the line adjusting mechanism can be activated to achieve more aesthetically pleasant output: *troff* can compute line breaks for a whole paragraph at once, using a variant of an algorithm originally developed by Donald Knuth and Michael Plass for the TEX system [7]. Interword spaces can be shrunk as an alternative to being expanded. Inter-letter spaces and letter shapes can be dynamically varied both for computing break points and for adjusting output lines; this is sometimes called ‘‘micro-typography’’, cf. e.g. the thesis by Hàn Thế Thành [8]. All paragraph formatting options can be arbitrarily combined.

A variety of international paper formats and hyphenation languages are supported by *troff.* The algorithm for the latter is derived from the respective one developed for TEX by Franklin Liang [9]; *troff* uses the implementation of *LibHnj* by Raph Levien. TEX hyphenation patterns can be converted to the format accepted by *troff.*

Many internal limitations of *troff* have been removed; most notably, *troff* can now set characters in fractional point sizes.

The *troff* language has been extended similarly as in the GNU version of *troff, groff* [10]; for example, names of requests, macros, strings, number registers, and fonts can consist of more than two characters. Although *Heirloom troff* is not completely compatible with *groff,* a special compatibility mode is provided, and documents prepared for *groff* can usually be processed without alteration.

As even the most basic printing devices are now capable of rastering PostScript documents (at least using conversion tools), and as PostScript and PDF viewer programs allow an accurate on-screen display of *troff* documents, there is only one area where *nroff* is still useful: the formatting of UNIX manual pages. The *Heirloom* version of *nroff* is thus specifically aimed at this task; it remains a separate program, is much smaller than *troff,* and is optionally able to run without external device description files. This makes it possible to use it to view manual pages even on small system distributions where the size of programs is an issue.

The Solaris version of *nroff* had already been updated to support input characters in arbitrary locales. The *Heirloom* version adds the ability to generate UTF-8 output. This extends the *nroff* character set by many mathematical and typographical characters on terminals capable of displaying them.

Although *Heirloom nroff* does of course not provide the typographical extensions made for *troff,* it includes the same language extensions. It is thus well able to cope with the tangle of manual page code which has been produced by application writers who have unfortunately become increasingly unaware of how to write well-styled *nroff* documents.

This manual as well as some of the pre- and post-processor commands have been derived from UNIX code released by Caldera [11], and from materials released by Lucent as parts of the *Plan 9* system [12].

Background to the Second Edition

troff was originally written by the late Joe Ossanna in about 1973, in assembly language for the PDP-11, to drive the Graphic Systems CAT typesetter. It was rewritten in C around 1975, and underwent slow but steady evolution until Ossanna's death late in 1977.

In 1979, Brian Kernighan modified *troff* so that it would produce output for a variety of typesetters, while retaining its input specifications. Over the decade from 1979 to 1989, the internals have been modestly revised, though much of the code remains as it was when Ossanna wrote it.

troff reads parameter files each time it is invoked, to set values for machine resolution, legal type sizes and fonts, and character names, character widths and the like. *troff* output is ASCII characters in a simple language that describes where each character is to be placed and in what size and font. A post-processor must be written for each device to convert this typesetter-independent language into specific instructions for that device.

The output language contains information that was not readily identifiable in the older output. In the newer language, the beginning of each page, line, and word is marked, so post-processors can do device-specific optimizations such as sorting the data vertically or printing it boustrophedonically, independent of *troff.*

Capabilities for graphics have been added: *troff* recognizes commands for drawing diagonal lines, circles, ellipses, circular arcs, and quadratic B-splines. There are also ways to pass arbitrary information to the output, unprocessed by *troff.*

A number of limitations have been eased or eliminated. A document may have an arbitrary number of fonts on any page (if the output device permits it, of course). Fonts may be accessed merely by naming them; ''mounting'' is no longer necessary. There are no limits on the number of characters. Character height and slant may be set independently of width.

The remainder of this document contains a description of usage and command-line options; a summary of requests, escape sequences, and pre-defined number registers; a reference manual; tutorial examples; and a list of commonly-available characters.

Acknowledgements (for the Second Edition, by Brian Kernighan)

Joe Ossanna's *troff* remains a remarkable accomplishment. For more than twenty years, it has proven a robust tool, taking unbelievable abuse from a variety of preprocessors and being forced into uses that were never conceived of in the original design, all with considerable grace under fire.

Recent versions of *troff* have profited from significant code improvements by Jaap Akkerhuis, Dennis Ritchie, Ken Thompson, and Molly Wagner. Andrew Hume, Doug McIlroy, Peter Nelson, and Ravi Sethi made valuable suggestions on the manual. I fear that the remaining bugs are my fault.

Acknowledgements (for the Heirloom Edition, by Gunnar Ritter)

The *troff* program as written by Joseph Ossanna and Brian Kernighan turned out to be an excellent base for a typesetting system containing approximately twice as many lines of code. As with the second edition, the design and the principal parts of *troff* have remained the same, despite of additions and modifications. This is particularly remarkable since much of it is now more than 40 years old, but serves its purpose as well as on day one.

I am grateful to Sun, Caldera, and Lucent for releasing the source code without which this project would not have been possible.

I wish to thank Brian Kernighan for the permission to adapt this manual.

Availability and Contact Information

The source code of the *Heirloom Documentation Tools* and accompanying materials are freely available from <http://n-t-roff.github.io/heirloom/doctools.html>. Comments and bug reports should be added at <https://github.com/n-t-roff/heirloom-doctools/issues> or send to <troff@arcor.de>.

References

[1] M. D. McIlroy, B. W. Kernighan (Eds.), *Unix Programmer's Manual,* Vol. 2, AT&T Bell Laboratories, Murray Hill, NJ 07974, 1979; <http://plan9.bell-labs.com/7thEdMan/index.html>.

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

[3] M. E. Lesk, ''Tbl — A Program to Format Tables'', in [1].

[4] B. W. Kernighan, ''A TROFF Tutorial'', in [1].

[5] J. F. Ossanna, ''Nroff/Troff User's Manual'', in [1].

[6] <http://www.opensolaris.org>

[7] D. E. Knuth, M. F. Plass, ''Breaking paragraphs into lines'', *Software—Practice and Experience*, Vol. 11, Issue 12 (1981), pp. 1119–1184; also in D. E. Knuth, *Digital Typography*, Stanford, 1999 (CSLI lecture notes no. 78), pp. 67–155.

[8] Hàn Thế Thành, *Micro-typographic extensions to the TEX typesetting system*, Masaryk University Brno, 2000.

[9] F. M. Liang, *Word Hy-phen-a-tion by Com-put-er*, Stanford University, CA 94305, Report No. STAN-CS-83-977, 1983.

[10] <http://www.gnu.org/software/groff>

[11] <http://www.tuhs.org/Archive/Caldera-license.pdf>

[12] <http://plan9.bell-labs.com/plan9>

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***
–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.
–n*N*	Number first generated page *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 include a ‘‘pause’’ code every *N* pages; its meaning, if any, depends on the output device.
–m*name*	Prepend the macro file **/usr/ucblib/doctools/tmac/***name* to the input *files*, or, if that file would not be accessible, **/usr/ucblib/doctools/tmac/tmac.***name*. If the environment variable **TROFFMACS** is set, its value is used instead of the default macro directory string, and no attempt is made to open *name* with the ‘‘tmac.’’ prefix. The value is prepended to *name* without inserting an additional slash as a directory separator, so it must either end with a slash itself or can be used to specify a file name prefix (as e.g. ‘‘tmac.’’).
–r*aN*	Register *a* (one-character) is set to *N*.
–r*a***=**N	Register *a* (may be more than one character) is set to *N*.
–d*aS*	String *a* (one-character) is set to *S*.
–d*a***=***S*	String *a* (may be more than one character) is set to *S*.
–i	Read standard input after the input files are exhausted.
–q	Invoke the simultaneous input-output mode of the **rd** request.
–x*N*	Set the extension level to *N* (cf. §1.1).
–z	Check syntax only; do not generate any output except for error messages.
	nroff Only
–e	Produce equally-spaced words in adjusted lines, using full terminal resolution.
–h	Use output tabs during horizontal spacing to speed output and reduce output character count. Tab settings are assumed to be every 8 nominal character widths.
–T*name*	Specify the name of the output terminal type. Currently defined names are **37** for the (default) Model 37 Teletype®, **lp** for any line printer or terminal without half-line capability, **tn300** for the GE TermiNet 300, **300S** for the DASI-300S, **300** for the DASI-300, and **450** for the DASI-450 (Diablo Hyterm). A special name, **locale**, is also supported. It generates UTF-8 output if permitted by the current LC_CTYPE locale, and has the same effect as **lp** otherwise.

troff Only

–f Refrain from feeding out paper and stopping the output device at the end of the run.

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

–F*path* Look in directory *path* for font information; the default is **/usr/ucblib/doctools/font/devps**.

–T*name* Specifies the output device. Currently defined names are **ps** for (default) PostScript output at 72 000 dpi, **psmed** for PostScript output at 3600 dpi, **pslow** for PostScript output at 432 dpi, **post** for PostScript output at 720 dpi with legacy fonts, and **html** for producing HTML output with the **dhtml** postprocessor.

–u*N* Set the emboldening amount, i.e. the number of times a character is printed to simulate bold output, to *N*.

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*[2] (for *nroff* and *troff* respectively), the table-construction preprocessor *tbl*[3], and *pic* and *grap* for various forms of graphics. A reverse-line postprocessor *col* 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. *col* can optionally also filter the backspace sequences which *nroff* emits for underlining and emboldening in order to produce a plain text file. Another option is the *ul* postprocessor which converts backspace sequences to underline and reverse video escape sequences for CRT terminals.

The **dpost** postprocessor has a special role as it translates *troff* intermediate output to PostScript, which is currently the only relevant target device format.

For example, in

tbl *files* **|** **eqn** **|** **troff** *options* **|** **dpost** >*output.ps*

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 *dpost,* which then writes PostScript code to *output.ps*.

The following options are currently supported with *dpost:*

Option	***Effect***
–e *encoding*	Set the PostScript encoding scheme. Possible values of *encoding* are: **0** using the PostScript **ashow** operator, with the same representation as previous versions of *dpost* **1** using **ashow**, integrating motion commands with text commands **2** using the PostScript **awidthshow** operator, computing space widths in PostScript **3** using **awidthshow**, computing space widths in *dpost* **4** using **ashow**, storing text positions as differences **5** like **4** but using a binary PostScript Level 2 representation The default is device-specific. **3** is the default with the high-resolution **ps** device; it usually produces the most efficient PostScript output and leads to the most compact PDF documents. With lower-resolution devices, **2** is preferred over **3** since the latter may produce incorrect alignment because of accumulated rounding errors. **0**, **1**, or **4** may be preferable if the text font contains many kerning pairs since **2** or **3** can result in less efficient or less compact output under these circumstances.

–M *marks* Print marks (in combination with the **trimat** *troff* request). Valid types of *marks* are: **cutmarks**, **registrationmarks**, **startargets**, **colorbars**, and **all**. Mark names can be abbreviated and combined by colons, e.g. **–Mcut:reg** will print cut marks and registration marks.

Request Summary

In the following table, the notation ±*N* in the *Request Form* column means that the forms *N*, +*N*, or –*N* are permitted, to set the parameter to *N*, increment it by *N*, or decrement it by *N*, respectively. Plain *N* means that the value is used to set the parameter. *Initial Values* separated by **;** are for *nroff* and *troff* respectively. In the *Notes* column,

B	Request normally causes a break. The use of ´ as control character (instead of **.**) suppresses the break function.
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.
T	*troff* only; no effect in *nroff.*
v,**p**,**m**,**u**	Default scale indicator; if not specified, scale indicators are *ignored*.

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
1. General Explanation				
.do *name*		ignored	–	Execute *name* in extension level 3.
.xflag *N*	1	ignored	–	Set the extension level permanently to *N*.
2. Font and Character Size Control				
.lc_ctype *name*		ignored	–	Set the LC_CTYPE locale.
.ps ±*N*	10 point	previous	E,T,**p**	Point size; also 'e_s'**\s**±*N*.
.fzoom *F Z*	1	ignored	P,T	Zoom font *F* by factor *Z*.
.ss *N* [*M*]	12/36 m	ignored	E,T	Space-character size set to *N*/36 em; sentence space *M*/36 em.
.cs *F N M*	off	–	P,T	Constant character space (width) mode (font *F*).
.bd *F N*	off	–	P,T	Embolden font *F* by *N*–1 units.
.bd S *F N*	off	–	P,T	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* [*file* [*supply*]]		ignored	P	Font position; mounts the font *file*.
.fps *map ...*		ignored	P,T	Mount a font with a special character map.
.feature *F* ±*name ...*		ignored	P,T	Control OpenType features.
.fallback *F A B ...*		ignored	P,T	Select the fallback sequence for font *F*.
.hidechar *F c d ...*		ignored	P,T	Hide the characters *c*, *d*, etc. from font F.
.spacewidth *N*		on	O,T	If *N*≠0, use the space width from the font metrics file.
.fspacewidth *F* [*N*]		ignored	O,T	Set the width of the space character in font *F* to *N*.
3. Page Control				
.pl ±*N*	11i	11i	**v**	Page length.
.papersize *media*		ignored	T,**u**	Set the paper size.
.mediasize *media*		ignored	T,**u**	Set the device media size.
.cropat *L T W H*		ignored	T,**p**	Set the "CropBox" page parameter for PDF generation.
.trimat *L T W H*		ignored	T,**p**	Set the "TrimBox" page parameter for PDF generation.
.bleedat *L T W H*		ignored	T,**p**	Set the "BleedBox" page parameter for PDF generation.
.bp ±*N*	*N*=1	–	B	Eject current page; next page number *N*.
.pn ±*N*	*N*=1	ignored	–	Next page number *N*.
.po ±*N*	0;1i	previous	**m**	Page offset.
.ne *N*	–	*N*=1v	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.

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
4. Text Filling, Adjusting, and Centering				
.br	–	–	B	Break.
.lsm *xx*	none	none	–	Leading space macro is *xx*.
.brp	–	–	B	Break and spread.
.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*; *c*=l,r,c,b,p,*none*
.na	adjust	–	E	No output line adjusting.
.padj *N*	off	on	–	Control paragraph-at-once adjustment globally.
.ce *N*	off	*N*=1	B,E	Center following *N* input text lines.
.rj *N*	off	*N*=1	B,E	Right-align following *N* input text lines.
.brnl *N*	off	*N*=∞	B,E	Break at end of next *N* input text lines.
.brpnl *N*	off	*N*=∞	B,E	Break and spread at end of next *N* input text lines.
.minss *N*	off	off	E,T	Minimum word space when adjusting lines.
.letadj *X U S Y V*		off	E,T	Dynamic letter spacing and reshaping when adjusting lines.
.sentchar *c...*	**.?!:**	off	E	Sentence-ending characters.
.transchar *c..*	**"')]*†**	off	E	Transparent characters for sentence-ending.
.track *F S N T M*		ignored	P,T,**p**	Static letter space tracking.
.kern *N*	1	1	P,T	Control pairwise kerning.
.fkern *F N*	1	1	P,T	Control the use of kerning tables from font *F*.
.kernpair *F c... G d... N*		ignored	P,T	Define a kerning pair.
.kernafter *F c... N d... M ...*			P,T	Add a constant amount of space after a character.
.kernbefore *F c... N d... M ...*			P,T	Add a constant amount of space before a character.
.lhang *F c... N d... M ...*		ignored	T	Hanging characters at left margin.
.rhang *F c... N d... M ...*		ignored	T	Hanging characters at right margin.
5. Vertical Spacing				
.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				
.ll ±*N*	6.5 i	previous	E,**m**	Line length.
.in ±*N*	*N*=0	previous	B,E,**m**	Indent.
.ti ±*N*	–	ignored	B,E,**m**	Temporary indent.
.pshape ±*I1* ±*L1* ±*I2* ±*L2* ...		off	E,**m**	Define the shape of the current paragraph in **ad p** mode.
7. Macros, Strings, Diversion, and Position Traps				
.de *xx yy*	–	*.yy*=**..**	–	Define or redefine macro *xx;* end at call of *yy*.
.am *xx yy*	–	*.yy*=**..**	–	Append to a macro.
.ds *xx string*	–	ignored	–	Define a string *xx* containing *string*.
.as *xx string*	–	ignored	–	Append *string* to string *xx*.
.lds *xx string*	–	ignored	–	Define local string *xx* containing *string*.
.substring *xx N* [*M*]		*M*=−1	–	Replace string *xx* by its substring between *N* and *M*.
.length *R string*		*R* set to 0	–	Store the length of *string* in register *R*.
.index *R xx string*		ignored	–	Store position where *string* occurs in *xx* in register *R*.
.chop *xx*	–	ignored	–	Remove the last character of *xx*.
.rm *xx*	–	ignored	–	Remove request, macro, or string.

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.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*.
.box *xx*	–	end	D	Divert output to macro *xx*, excluding a partially filled line.
.boxa *xx*	–	end	D	Divert and append to *xx*, excluding a partially filled line.
.unformat *xx*	–	ignored	–	Strip line break information from diversion *xx*.
.asciify *xx*	–	ignored	–	All characters in diversion *xx* changed to plain text.
.wh *N xx*	–	–	**v**	Set location trap; negative is w.r.t. page bottom.
.ch *xx N*	–	–	**v**	Change trap location.
.dwh *N xx*	–	–	D,**v**	Set location trap in current diversion.
.dch *xx N*	–	–	D,**v**	Change trap location in current diversion.
.dt *N xx*	–	off	D,**v**	Set a diversion trap.
.vpt *N*	1	ignored	–	Enable or disable vertical position traps.
.it *N xx*	–	off	E	Set an input-line count trap.
.itc *N xx*	–	off	E	Input-line count trap ignoring **\c**.
.return	–	–	–	Return from the current macro.
.shift *N*	–	1	–	Shift the arguments to the current macro.
.als *yy xx*	–	–	–	*yy* is created as an alias for *xx*.
.blm *xx*	none	none	–	Blank line macro is *xx*.
.em *xx*	none	none	–	End macro is *xx*.
.recursionlimit *N M*		–	–	Set the maximum stack depth.

8. Number Registers

.nr *R* ±*N M*		–	**u**	Define and set number register *R*; auto-increment by *M*.
.nrf *R* ±*F G*		–	**u**	Define and set floating-point register *R*; auto-increment by *G*.
.lnr *R* ±*N M*		–	**u**	Define and set local number register *R*.
.lnrf *R* ±*F G*		–	**u**	Define and set local floating-point register *R*.
.af *R c*	arabic	–	–	Assign format to register *R* (*c*=**1**, **i**, **I**, **a**, **A**).
.rr *R*	–	–	–	Remove register *R*.
.rnn *R S*	–	–	–	Rename register *R* to *S*.
.aln *S R*	–	–	–	Register *S* is created as an alias for *R*.

9. Tabs, Leaders, and Fields

.ta *Nt* ...	8 n; 0.5 i	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.
.ecs	\	–	–	Save escape character.
.ecr	\	–	–	Restore saved escape character.
.escoff *esc* ...	on	ignored	–	Disable escapes *esc* ... .
.escon *esc* ...	on	ignored	–	Enable escapes *esc* ... .
.lg *N*	–; on	on	T	Ligature mode on if *N*>0.
.flig *F string c* ...		ignored	T	Define the ligatures in font *F*.
.fdeferlig *F string* ...		ignored	T	Defer ligature building for the first character of *string*.
.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*.

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
.c2 *c*	´	´	E	Set nobreak control character to *c*.
.tr *abcd....*	none	–	O	Translate *a* to *b*, etc. on output.
.trin *abcd....*	none	–	O	Translate but retranslate with **asciify**.
.trnt *abcd....*	none	–	O	Translate but not on transparent lines.
.ftr *F abcd....*	none	–	P,T	Font-specific **tr**.
.char *c string*		ignored	–	Define character *c* to *string*.
.fchar *c string*		ignored	T	Define fallback for character *c* to *string*.
.rchar *c...*		ignored	–	Remove character definitions for *c....*
.utf8conv *c1 c2*		ignored	–	*nroff* UTF-8 locale character translation.
.output *string*		ignored	–	Write *string* directly to intermediate output.

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

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

.connectchar *c..*	**"\(ru\(ul\(rn**	off	E	Connected characters for line drawing.

13. Hyphenation.

.nh	hyphenate	–	E	No hyphenation.
.hy *N*	hyphenate	hyphenate	E	Hyphenate; *N* = mode.
.hylang *name*	off	off	E	Set the hyphenation language.
.shc *c*	-	-	E	Set the soft hyphenation character.
.hcode *abcd...*		–	E	Hyphenation code of *a* is *b*, etc.
.hylen *N*	5	5	E	Hyphenate only words of at least *N* characters in length.
.hlm *N*	off	off	E	Maximum number of consecutive hyphenated lines.
.hypp *N M L*	0 0 0	0 0 0	E	Define hyphenation penalties for **ad p** mode.
.breakchar *c.*	**-—**	off	E	Optional line break characters.
.nhychar *c...*	**-—**	off	E	Hyphenation-inhibiting characters.
.hc *c*	**\%**	**\%**	E	Hyphenation indicator character *c*.
.hw *word ...*	–	ignored	–	Add words to hyphenation dictionary.

14. Three-Part Titles.

.tl *´left´center´right´*		–	–	Three part title; delimiter may be any character.
.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.

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 [sic], accept *anything*.
.if f*F anything*		–	**u**	If floating-point expression *F* > 0, accept *anything*.
.if !f*F anything*		–	**u**	If floating-point expression *F* ≤ 0 [sic], 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.
.while *c anything*		–	–	Execute *anything* while *c* (like **if**) is true.
.break *n*	–	1	–	Break out of *n* nested **while** loops.

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.continue *n*	–	1	–	Continue at the *n*-th nested **while** loop.

17. Environment Switching.

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.ev *name*	*name*=0	previous	–	Environment switched (*push down*).
.evc *name*		–	–	Copy environment *name* to the current environment.

18. Insertions from the Standard Input

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

19. Input/Output File Switching

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.so *filename*		–	–	Switch source file *(push down)*.
.pso *string*		–	–	Execute *string* and read its output.
.nx *filename*		end-of-file	–	Next file.
.sy *string*		–	–	Execute program *string*. Output not interpolated.
.pi *string*		–	–	Pipe output to program *string*.
.cf *filename*		–	–	Copy file contents to *troff* output.
.open *stream filename*		ignored	–	Open *filename* as *stream*, truncating.
.opena *stream filename*		ignored	–	Open *filename* as *stream*, appending.
.write *stream text*		ignored	–	Write *text* to file *stream*.
.writec *stream text*		ignored	–	Write *text* without terminating newline.
.writem *stream xx*		ignored	–	Write contents of string, macro, or diversion *xx*.
.close *stream*		–	–	Close the file *stream*.

20. Miscellaneous

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.mc *c N*	–	off	E,**m**	Set margin character *c* and separation *N*.
.lpfx *string*	off	off	E	Set line prefix to *string*.
.tm *string*	–	newline	–	Print *string* on terminal (standard error).
.tmc *string*	–	newline	–	Print *string* without newline on terminal.
.nop *remainder of line*		–	–	Use *remainder of line* as input.
.chomp *remainder of line*		–	–	Use *remainder of line without newline* as input.
.ab *string*	–	newline	–	Print *string* on standard error, exit program.
.ig *yy*	–	*.yy*=**..**	–	Ignore till call of *yy*.
.lf *N f*		–	–	Set input line number to *N* and filename to *f*.
.pm *t*	–	all	–	Print macro names and sizes; if *t* present, print only total of sizes.
.fl	–	–	B	Flush output buffer.

21. Output and Error Messages, Debugging

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.warn ±*bits*|*name*		*w*	–	Control warning messages.
.spreadwarn *N*		toggle	**m**	Spread limit warning.
.errprint *string*		newline	–	Print *string* like an error message.
.watch *xx*	off	ignore	–	Notify on change of string or macro *xx*.
.unwatch *xx*	off	ignore	–	Disable notification for string or macro *xx*.
.watchlength *N*		ignore	–	On change, report contents up to length *N*.
.watchn *R*	off	ignore	–	Notify on change of register *R*.
.unwatchn *R*	off	ignore	–	Disable notification for register *R*.

22. Color Support

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.CL *color text*		RGB black	–	Print *text* in *color*.*

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

23. Picture Inclusion

.psbb *filename*		–	–	Read the PostScript bounding box from *filename*.
.BP *source height width position offset flags label*				Define a frame and place a picture in it.*
.EP	–	–	–	End a picture started by **.BP**.*
.PI *source height***,***width***,***yoffset***,***xoffset flags*				Low-level picture inclusion.*

24. Special Features for PDF Documents

25. *groff* Compatibility

.cp *N*	off	–	–	Enable *groff* compatibility mode.
.mso *name*	–	ignored	–	Include the macro package *name*.*

26. Output Language

27. Device and Font Description Files

* Defined as a macro in an external package; refer to the detailed description on how to include it.

Alphabetical Request and Section Number Cross Reference

Escape Sequences for Characters, Indicators, and Functions

Section Reference	*Escape Sequence*	*Meaning*
10.1	\\	\ (to prevent or delay the interpretation of \ )
2.1	\´	´ (acute accent); equivalent to **\(aa**
2.1	\`	` (grave accent); equivalent to **\(ga**
2.1	\–	– Minus sign in the *current* font
12.4	_	_ (underrule character); equivalent to **\(ul**
7	\.	Period (dot) (see **de**)
4.1	\(space)	Unpaddable space-size space character
4.1	\~	Paddable no-break space character
11.1	\0	Digit width space
11.1	\\|	1/6 em narrow space character (zero width in *nroff*)
11.1	\^	1/12 em half-narrow space character (zero width in *nroff*)
4.1	\&	Non-printing, zero width character
4.1	\)	Transparent non-printing zero width character
10.6	\!	Transparent line indicator
10.8	\"	Beginning of comment
10.8	\#	Comment including newline
7.3	\$*n*,\$(*nn*,\$[*nnn*]	Interpolate argument *n*, *nn*, or *nnn*
7.3	\$*	Interpolate all arguments separated by spaces
7.3	\$@	Interpolate all arguments in double quotes
7.3	\$0	Interpolate name of current macro or string
13	\%	Default optional hyphenation character
2.1	\(*xx*	Character named *xx*
2.1	\[*xxx*]	Character named *xxx*
7.1	*x,*(xx,*[xxx]	Interpolate string *x*, *xx*, or *xxx*
7.1	*[*xxx arg* ...]	Interpolate string *xxx* with arguments *arg* ...
13	\:	Optional line-break character
10.2	\;	Ligature suppressor
17	\@{,\@}	Inline environment push/pop
9.1	\a	Non-interpreted leader character
24.6	\A´*string*´	Anchor definition
12.3	\b´*abc...*´	Bracket building function
1.4	\B'*string*'	Test if *string* is a numerical expression
4.2	\c	Interrupt text processing
2.1	\C´*xxx*´	Character named *xxx*
11.1	\d	Forward (down) 1/2 em vertical motion (1/2 line in *nroff*)
12.5	\D´*c...*´	Draw graphics function *c* with parameters ...; *c*=**l**,**c**,**e**,**a**,~
10.1	\e	Printable version of the *current* escape character
10.1	\E	Escape character, not interpreted in *copy mode*
2.2	\f*x*,\f(*xx*,\f[*xxx*],\f*N*	Change to font named *x*, *xx*, or *xxx*, or position *N*
8	\g*x*,\g(*xx*,\g[*xxx*]	Format of number register *x*, *xx*, or *xxx*
11.1	\h´*N*´	Local horizontal motion; move right *N (negative left)*
2.3	\H´*N*´	Height of current font is *N*
1.1	\I'*string*'	Test if *string* is a valid identifier name
4.1	\j´±*N*´	Penalty for breaking a line after the current word is *N*
4.1	\J´±*N*´	Default line breaking penalty is *N*
11.3	\k*x*,\k(*xx*,\k[*xxx*]	Mark horizontal *input* place in register *x, xx*, or *xxx*
12.4	\l´*Nc*´	Horizontal line drawing function (optionally with *c*)
12.4	\L´*Nc*´	Vertical line drawing function (optionally with *c*)
8	\n*x*,\n(*xx*,\n[*xxx*]	Interpolate number register *x*, *xx*, or *xxx*
2.	\N´*N*´	Character number *N* on current font
12.1	\o´*abc...*´	Overstrike characters *a, b, c, ...*

4.1	**\p**	Break and spread output line
7.5	**\P***x*,**\P(***xx*,**\P[***xxx*]	Output-line trap *x*, *xx*, or *xxx*
11.1	**\r**	Reverse 1 em vertical motion (reverse line in *nroff*)
8	**\R**´*R* ±*N*´	Set number register *R* to ±*N*
2.3	**\s***N*,**\s**±*N*, **\s**´±*N*´,**\s**±´*N*´, **\s[**±*N*],**\s**±[*N*]	Point-size change function
2.2	**\S**´*N*´	Slant output *N* degrees
9.1	**\t**	Non-interpreted horizontal tab
24.6	**\T**´*string*´	Intra-document link definition
11.1	**\u**	Reverse (up) 1/2 em vertical motion (1/2 line in *nroff*)
2.1	**\U**´*X*´	Character at Unicode position U+*X*
11.1	**\v**´*N*´	Local vertical motion; move down *N (negative up)*
20	**\V***x*,**\V(***xx*,**\V[***xxx*]	Environment variable *x*, *xx*, or *xxx*
11.2	**\w**´*string*´	Interpolate width of *string*
24.6	**\W**´*string*´	URI link definition
5.2	**\x**´*N*´	Extra line-space function *(negative before, positive after)*
10.7	**\X**´*string*´	Output *string* as device control function
10.7	**\Y***x*,**\Y(***xx*,**\Y[***xxx*]	Output contents of macro *x*, *xx*, or *xxx* as device control function
12.2	**\z***c*	Print *c* with zero width (without spacing)
12.2	**\Z**´*string*´	Print *string* with zero width and height
16	**\{**	Begin conditional input
16	**\}**	End conditional input
10.7	\(newline)	Concealed (ignored) newline
–	*c*	*c*, any character *not* listed above

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

Predefined General Number Registers

Section Reference	***Register Name***	***Description***
3	**%**	Current page number.
–	**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).
–	**hours**	Hours portion of current local time (0–23).
11.3	**hp**	Current horizontal place on *input* line.
15	**ln**	Output line number.
–	**minutes**	Minutes portion of current local time (0–59).
–	**mo**	Current month (1–12).
4.1	**nl**	Vertical position of last printed text base-line.
11.2	**rsb**	Visual depth of string below base line (generated by *width* function).
11.2	**rst**	Visual height of string above base line (generated by *width* function).
11.2	**sb**	Depth of string below base line (generated by *width* function).
–	**seconds**	Seconds portion of current local time (0–60).
11.2	**st**	Height of string above base line (generated by *width* function).
–	**year**	Current year.
–	**yr**	Current year minus 1900.
25.1	**.g**	Current *groff* compatibility mode (0=off).

Predefined Read-Only Number Registers

Section Reference	*Register Name*	*Description*
19	**$$**	Process id of *nroff* or *troff.*
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.*
5.2	**.a**	Post-line extra line-space most recently utilized using **\x**´*N*´.
5.4	**.ascender**	Ascender of current font and point size.
2.3	**.b**	Emboldening level.
13	**.breakchar**	Current optional line break characters.
4.1	**.brnl**	Remaining number of lines with break at newline.
4.1	**.brpnl**	Remaining number of lines with break and spread at newline.
–	**.c**	Number of *lines* read from current input file.
11.2	**.cdp**	Visual depth below base line of previous character.
4.1	**.ce**	Remaining number of lines to be centered.
11.2	**.cht**	Visual height above base line of previous character.
12.4	**.connectchar**	Current characters connected for line drawing [sic].
7.4	**.d**	Current vertical place in current diversion; equal to **nl**, if no diversion.
4.1	**.defpenalty**	Default line breaking penalty.
5.4	**.descender**	Descender of current font and point size.
7.4	**.dilev**	Current diversion level.
17	**.ev**	Name of current environment [sic].
2.2	**.f**	Current font as physical quadrant (1-255).
2.2	**.fp**	Next unused physical font quadrant.
2.3	**.fzoom**	Current font zoom factor (may be a decimal fraction).
20	**.F**	Current input file name [sic].
4	**.h**	Text base-line high-water mark on current page or diversion.
13	**.hlc**	Current number of consecutive hyphenated lines.
13	**.hlm**	Maximum number of consecutive hyphenated lines.
13	**.hy**	Current hyphenation flags.
13	**.hylang**	Current hyphenation language [sic].
13	**.hylen**	Current minimum hyphenation word length.
13	**.hypp**	Penalty for hyphen in **ad p** mode.
13	**.hypp2**	Penalty for consecutive hyphens in **ad p** mode.
13	**.hypp3**	Penalty for hyphenating the last word of a paragraph in **ad p** mode.
11.1	**.H**	Available horizontal resolution in basic units.
6	**.i**	Current indent as set by **in**.
6	**.in**	Current indent including temporary indent, if any.
4.2	**.int**	Non-zero if the previous line was interrupted with **\c**.
4	**.j**	Current **ad** mode.
4.1	**.k**	Current output horizontal position.
4.2	**.kc**	Output horizontal length of interrupted word, if any.
6	**.l**	Current line length.
14	**.lt**	Current title length.
2.1	**.lc_ctype**	Current LC_CTYPE locale [sic].
4.1	**.letss**	Current dynamic letter space threshold.
20	**.lpfx**	Current line prefix [sic].
4.1	**.lshmin**	Current minimum dynamic letter shape (may be a decimal fraction).
4.1	**.lshmax**	Current maximum dynamic letter shape (may be a decimal fraction).
4.1	**.lspmin**	Current minimum dynamic letter space (may be a decimal fraction).
4.1	**.lspmax**	Current maximum dynamic letter space (may be a decimal fraction).
5.1	**.L**	Current **ls** value.
4.1	**.minss**	Current minimum space size.

4	**.n**	Length of text portion on previous output line.
5	**.ns**	Non-zero if no-space mode is active.
13	**.nhychar**	Current hyphenation-inhibiting characters.
3	**.o**	Current page offset.
3	**.p**	Current page length.
4	**.padj**	Current paragraph-at-once global setting.
3	**.pn**	Number of next page.
2.3	**.ps**	Current point size in units.
2.3	**.psr**	Last requested point size in units.
4.1	**.rj**	Remaining number of lines to be right-aligned.
2.3	**.s**	Current point size (may be a decimal fraction).
13	**.shc**	Current soft hyphenation character [sic].
4.1	**.sentchar**	Current sentence-ending characters [sic].
2.3	**.sr**	Last requested point size (may be a decimal fraction).
2	**.ss**	Current space size.
2	**.sss**	Current sentence space size.
9.1	**.S**	Current tab stops such that they can be passed back to **ta**.
7.5	**.t**	Distance to the next trap.
9.1	**.tabs**	Current tab stops such that they can be passed back to **ta**.
4.1	**.transchar**	Current characters transparent for sentence-ending [sic].
–	**.T**	Set to 1 in *nroff,* if **–T** option used; always 0 in *troff.*
4.1	**.u**	Equal to 1 in fill mode and 0 in nofill mode.
5.1	**.v**	Current vertical line spacing.
7.5	**.vpt**	Vertical position traps enabled (1) or disabled (0).
11.1	**.V**	Available vertical resolution in basic units.
11.2	**.w**	Width of previous character.
21	**.warn**	Currently activated warning categories.
4.1	**.x**	Remaining horizontal space on current output line.*
1.1	**.X**	Current extension level.
6	**.y**	Current indent including temporary indent, if any.*
7.4	**.z**	Name [sic] of current diversion.
4.1	**lsn**	Number of leading spaces of a input line in fill mode.
4.1	**lss**	Horizontal space corresponding to a line with leading spaces in fill mode.

* The **.x** and **.y** registers had been described as "reserved version-dependent registers" in previous editions. Their semantics have actually been as described from 7th Edition *troff* on.

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 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 esthetic 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 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*, a left-parenthesis-introduced, two-character name as in **\n(***xx*, or a left-bracket-introduced, multiple character name as in **\n[***xxx*].

In traditional *troff,* only one and two character names were permitted for request, macro, string, number register, and font names. Heirloom *troff* can accept names containing a (nearly) arbitrary number of ASCII characters. Escape **\I'***string*' can by used to test if *string* is a valid identifier name. By default, request and macro names are still required to contain at most two characters for compatibility reasons. The **–x** command line option, the **do** request, or the **xflag** request make the longer names available. The current extension level is available in the **.X** register.

Four levels of extension availability are currently defined:

0 disables all extensions except for locale-dependent input and Type 1/OpenType/TrueType font selection using **fp**. Since the **do** request is not available at this level, it is not possible to change to another extension level again. It is most useful to print unmaintained documents for which any adaption would be too tedious.

1 enables extensions except for direct access to long names, i.e., **.abcde** will be interpreted as request **ab** with argument **cde**, and ***[xyz]** refers to the string named **[** followed by the text **xyz]**. Long names can be accessed using the **do** request, e.g. **.do de abcde**, **.do if 1 *[xyz]**. This level is the default.

2 enables direct access to long names on request lines and escape sequences, i.e., **.abcde** will be interpreted as macro **abcde**, and ***[xyz]** refers to the string named **xyz**. If an undefined long name is read, its first two characters are interpreted as a short request using the remaining characters as argument. So if a macro **abcde** is defined, **.abcde** will execute it, but otherwise, **.abcde** continues to execute **ab**. String and number register references are only interpreted if they start in the first two characters of a name, i.e. no string interpretation is performed on input **.ab*(xy**.

3 ignores undefined long requests even if they form a prefix of a short request, and interprets string and number register references in any position of a name.

It is recommended to execute **.do xflag 3** at the beginning of new *troff* programs that need not rely on any existing code.

1.2. Formatter and device resolution. *troff* internally stores and processes dimensions in units that correspond to the particular device for which output is being prepared; values from 300 to 72 000/inch are typical. See §27. *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 the actual horizontal/vertical resolution of the output device indicated by the **–T** option (default **ps**). *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 appended scale indicators shown in the following table, where *S* is the current type size in points, *V* is the current vertical line spacing in basic units, and *C* is a *nominal character width* in basic units.

Scale Indicator	Meaning	Number of basic units *troff –Tps*	*nroff*
i	Inch	72000	240
c	Centimeter	72000×50/127	240×50/127
P	Pica = 1/6 inch	12000	240/6
m	Em = *S* points	*S*	*C*
n	En = Em/2	*S*/2	*C, same as Em*
M	1/100 Em	*S*/100	*C*/100
p,**z**	Point = 1/72 inch	1000	240/72
u,**s**	Basic unit	1	1
t	Printer's point	72000×100/7227	240×100/7227
T	Printer's pica	72000×400/2409	240×400/2409
D	Didot point	72000×24/1621	240×24/1621
C	Cicero	72000×288/1621	240×288/1621
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 **m** for the horizontally-oriented requests and functions **ll**, **in**, **ti**, **ta**, **lt**, **po**, **mc**, **spreadwarn**, **\h**, **\l**, and horizontal coordinates of **\D**; **v** for the vertically-oriented requests and functions **pl**, **wh**, **ch**, **dt**, **sp**, **sv**, **ne**, **rt**, **\v**, **\x**, **\L**, and vertical coordinates of **\D**; **p** for the **vs**, **papersize**, **mediasize**, **trimat**, **bleedat**, and **cropat** requests; and **u** for the requests **nr**, **nrf**, **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, except for floating-point computations with **nrf** and **if f**. Exponential notation as in '1**e**+9' or '1**e–**10' is supported.

The *absolute position* indicator **|** may be prepended to a number *N* to generate the distance to the vertical or horizontal place *N*. For vertically-oriented requests and functions, **|***N* becomes the distance in basic units from the current vertical place on the page or in a *diversion* (§7.4) to the the vertical place *N*. For *all* other requests and functions, **|***N* becomes the distance from the current horizontal place on the *input* line to the horizontal place *N*. For example,

.sp |3.2c

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

1.4. Numerical expressions. Wherever numerical input is expected an expression involving parentheses, the arithmetic operators +, –, **/**, ∗, **%** (mod), the logical operators **<**, **>**, **<=**, **>=**, **=** (or **==**), **<>** (not equal), **&** (and), **:** (or), and the functions *a***<?***b* (minimum of *a* and *b*), *a***>?***b* (maximum of *a* and *b*), and **(***c***;***e***)** (evaluate the expression *e* using *c* as default scale indicator, or ignoring scaling indicators if *c* is omitted) may be used. Except where controlled by parentheses, evaluation of expressions is left-to-right; there is no operator precedence. Spaces are ignored if they occur within parentheses; any other non-numeric character terminates the expression. 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.

The **\B**'*string'* escape sequence checks whether *string* is a valid numerical expression and evaluates to "1" if it does and to "0" otherwise.

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.

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
.do *name*		ignored	–	Execute the request or macro *name* in extension level 3 and restore the previous level afterwards.
.xflag *N*	1	ignored	–	Set the extension level permanently to *N*.

2. Font and Character Size Control

2.1. Character set. The basic *troff* character set is defined by a description file specific to each output device (§27). There are normally several regular fonts and one or more special fonts. The basic character set is shown in the attached Table I. All ASCII characters are included, with some on the Special Font. With three exceptions, the 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 attached 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 characters **** and **"** can also be referred to as **\(rs** and **\(dq**, respectively; this form allows to completely avoid their interpretation as argument delimiter or escape character. In traditional *troff,*, the ASCII characters @, **#**, **"**, ´, `, **<**, **>**, ****, **{**, **}**, ˜, ˆ, and _ existed only on the Special Font and were printed as a 1-em space if that Font was not mounted. With the **pslow** device, these characters print in the *Times* font by default regardless of the current font (but see the **fps** request below). With other devices, these characters are taken from the current font.

With Type 1, OpenType, and TrueType fonts, *troff* allows to access all named PostScript characters of the current font and of those in the **fallback** sequence in the forms **\[***name***]** or **\C'***name***'**.

troff internally converts non-ASCII characters of the current LC_CTYPE locale to named PostScript characters once they are read in regular (not *copy*) mode. If the current font is an OpenType or a TrueType font and contains a custom Unicode mapping table, the input character is looked up in that table first. Otherwise, a default table is used. A character that is not present in the current font is searched using the **fallback** sequence first, then in the special fonts. If the character cannot be found, it is discarded. Characters for which no name is known are replaced by spaces.

The **\N'***n***'** escape sequence has historically been available to refer to character *n* of the current font. It is still accepted, but its use is discouraged with Type 1, OpenType, and TrueType fonts since the arrangement of character in font tables is performed at run-time and may change with future versions of *troff.* When register **.g** is set to **1** 8-bit character codes can also be accessed with **\[char***n***]** where $32 \le n \le 255$.

nroff has an analogous, but different, mechanism for defining legal characters and how to print them. By default all characters are valid. There are such 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. In a UTF-8 locale, combining characters are processed. The characters ´, `, and _ print as themselves.

Both *nroff* and *troff* allow references to specific Unicode characters with the **\U'***X***'** escape sequence; it causes the character at position U+*X* to be printed (*X* is a hexadecimal number). For *troff,* it is required that this character is available in one of the fonts mounted at this point. As an example, \U'20AC' prints the Euro character €. When register **.g** is set to **1** Unicode charactes can also be accessed with **\[u***XXXX***]** where *XXXX* is a four digit hexadecimal number.

2.2. Fonts. troff begins execution by reading information for a set of defaults fonts, said to be *mounted;* conventionally, the first four are Times Roman (**R**), Times Italic (**I**), Times Bold (**B**), and Times Bold Italic (**BI**), and the last is a Special font (**S**) containing miscellaneous characters. The set of fonts and positions is determined by the device description file, described in §27.

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*, **\f[***xxx***]**, or **\f***N* where *x*, *xx*, and *xxx* 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 as if they were physically part of the current font. The Special font may actually be several fonts; the name **S** is reserved and is generally used for one of these. The **fallback** request sets a font-specific sequence of additional fonts that are searched for missing characters.

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. The current font is available (as numerical position) in the read-only number register **.f**; the **.fp** register holds the next unused position.

A request for a named but not-mounted font is honored if the font description information exists. In this way, there is no limit on the number of fonts that may be printed in any part of a document. Mounted fonts may be handled more efficiently, and they may be referred to by their mount positions, but there is no other difference. Mention of an unmounted font loads it temporarily at font position zero, which serves as a one-font cache.

The function **\S***'±N'* causes the current font to be slanted by ±*N* degrees. Not all devices support slanting.

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

2.3. Character size. Character point sizes available depend on the specific output device; a typical (historical) set of values is 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. Modern output devices such as the default **devps** usually allow the point size to be set to arbitrary values, including fractional points. 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**±*N* (1≤*N*≤8) to increment/decrement the size by *N*; **\s0** restores the *previous* size. On historical output devices, requested point size values that are between two valid sizes yielded the larger of the two.

Note that through an accident of history, a construction like **\s39** is parsed as size 39, and thus converted to size 36 (given the sizes above), while **\s40** is parsed as size 4 followed by **0**. The forms **\s(***nn*, **\s**±**(***nn*, **\s´**±*nn´*, **\s**±*´nn´*, **\s[**±*nn***]**, and **\s**±**[***nn***]** permit specification of sizes that would otherwise be ambiguous.

The current size (in points) is available in the **.s** register. Note that this may be a decimal fraction if the current point size is not an integer. The **.ps** register stores the current size in units. The **.sr** and **.psr** registers store the requested point size in points and units, respectively; it is identical to the current size on modern output devices. *nroff* ignores type size control.

The function **\H***´±N'* sets the height of the current font to *N*, or increments it by +*N*, or decrements it by –*N*; if *N*=0, the height is restored to the current point size. In each case, the width is unchanged. Not all devices support independent height and width for characters.

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
.lc_ctype *name*		ignored	–	Set the LC_CTYPE locale to *name*. The default is the value of the LC_CTYPE environment variable. This request is useful to specify an input character set regardless of environment influences. The current value is available in the **.lc_ctype** number register.
.ps ±*N*	10 point	previous	E,**p**	Point size set to ±*N*. Alternatively imbed **\s***N*, **\s**±*N*, or **\s**´*N*´. Any positive size value may be requested on modern devices. If invalid, the next larger valid size will result on traditional devices, with a maximum of 36. A paired sequence +*N*, –*N* will work because the previous requested value is also remembered. Ignored in *nroff.*
.fzoom *F Z*	1	ignored	P,T	Zoom font *F* by factor *Z*, which is a positive floating point number. This request is useful for adjusting fonts with different visual sizes but identical nominal points; the zoom is thus not applied to characters from another font that have been found by the **fallback** sequence or by the special font mechanism. The current value of the current font is available in the **.fzoom** number register.
.ss *N* [*M*]	12/36 m	ignored	E,T	Space-character size is set to *N*/36 ems. This size is the minimum word spacing in adjusted text. The optional second argument sets the space that is added between sentences to *M*/36 ems. Ignored in *nroff.* The number registers **.ss** and **.sss** contain the current values of *N* and *M*, respectively.
.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 emboldening value *N* is in the **.b** register. This paragraph is printed with *.bd R 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*. 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, and the name **S** for the special font.

.fp *N F* [*file* [*supply*]] — ignored — P

Font position. This is a statement that a font named *F* is mounted on position *N*. With two arguments, it is a fatal error if *F* is not known as a legacy *troff* font file (§27.2).

With three or more arguments, font metrics are read from the given *file*, which must be in Type 1, OpenType, or TrueType format. If the **TROFFONTS** environment variable is set, each of the colon-separated directories in it is tested for the files *file.afm*, *file.otf*, *file.ttf*, or, if the *file* argument has an **.afm**, **.otf**, or **.ttf** extension itself, for *file*. Otherwise, the font is loaded from the file **/usr/ucblib/doctools/font/devps/***file***.afm**.

The *N* argument specifies a register on which the font is to be mounted (1≤*N*≤255). If it is zero, the font is mounted on the next free position (not on position zero). At most 255 fonts may be mounted simultaneously; it is possible to use more than 255 fonts in a document by reusing font registers.

The font is then available with **.ft F**, **\fF**, etc. **F** may be freely chosen, and may consist of more than two characters.

F may be the name of a previously mounted font, such as **R**. In this case, the same *N* register must be reused. For best compatibility with conventional *troff* usage, it is recommended that the base fonts of a document are mounted as "1 R", "2 I", "3 B", "4 BI". The initial fonts on positions 9 (**S1**) and 10 (**S**) should not be changed, as they contain special metrics for drawing commands.

If the optional *supply* argument is present, glyph data is included in the generated PostScript file. If *supply* is one of **otf**, **pfb**, **pfa**, **ttf**, or **t42**, the file *supply.pfb* (or likewise) is searched in the directories in **TROFFONTS** first as described for the AFM file above, and if it is not found there, in **/usr/ucblib/doctools/font/devps/***supply***.pfb** (or likewise). *supply* may also be the basename of a file like *file* above.

The *filename* may be surrounded by double-quotes to permit imbedded space characters.

.fps *map ...* — ignored — P,T

Mount a font with a special character map. By default, special characters like **\(*a** are not assigned when a font is mounted even if a matching named PostScript character (like **\[alpha]**) would have been available. The following special character maps exist:

math	mathematical characters like ≤ ∪ → ∞
greek	greek characters like α β γ Α Β Γ
punct	the characters \(or – \` \´ " # < > @ \ ^ |
large	parts of large characters like ⎤ ⎬ ⎣ ⎨

The **punct** map is used by default for the **ps** and **psmed** devices.

The remaining arguments are handled as described for **fp**.

.feature *F* ±*name ...* — ignored — P,T

Enable (+) or disable (–) the OpenType feature *name* in font *F*. Only OpenType features that result in context-insensitive single-character substitutions are supported. Typical features are **onum** to enable old-style numerals, or **smcp** to enable small capitals.

.fallback *F A B* ...	ignored	P,T	Select the fallback sequence for font *F*. If the current font is *F* and a character is not found, font *A* is searched first, then font *B*, etc. If the character still has not been found, it is searched for in the Special Font, then in the fonts mounted at positions 0, 1, and so forth.
.hidechar *F c d* ...	ignored	P,T	Hide the characters *c*, *d*, etc. from font F. If the characters appear in input afterwards, they are searched in other fonts using the fallback sequence. This is useful e.g. for combining characters from a regular and a Type 1 expert font.
.spacewidth *N*	on	O,T	If $N \neq 0$, use the space width from the font metrics file. The space width otherwise defaults to 1/3 em for variable-width fonts, or to the width of the **space** character for monospaced fonts. With this request, the space width is set to the width of the **space** character as obtained from the font metrics file for variable-width fonts too.
.fspacewidth *F* [*N*]	ignored	O,T	Set the width of the space character in font *F* to *N*, which is given in units of 1/72000 of an inch multiplied by the current point size or 1/1000 of an em (as in AFM kerning pair definitions). If only one argument is present, the space width is set to the width of the **space** character as obtained from the font metrics file. When the width of spaces in output is actually computed, the space size as set by **ss** is also taken into account. The space width as defined above is used directly with **.ss 12**, which is the default. Otherwise, it is multiplied by the **ss** setting divided by 12.

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 physical limitations on *troff* and *nroff* output are device dependent.

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
.pl $\pm N$	11 in	11 in	**v**	Page length set to $\pm N$. The current page length is available in the **.p** register.
.papersize *media*		ignored	T,**u**	Set the paper size. *media* can be one of **executive**, **letter**, **legal**, **ledger**, **tabloid**, **a***N*, **b***N*, **c***N*, or two numbers giving the width and height directly. The page length is set from these parameters, the page offset and line length are adjusted as needed, and the post-processor is informed about the page metrics for its internal calculations. The last action is the real reason why this request exists; without it, PostScript printers may displace the pages of the document. The default are **letter** measurements. This request should be used early in a document; if it is given multiple times, the last one will setup the device.

.mediasize *media*		ignored	T,**u**	Set the device media size. *media* can be one of **executive**, **letter**, **legal**, **ledger**, **tabloid**, **a***N*, **b***N*, **c***N*, or two numbers giving the width and height directly. This request performs the same actions as **papersize** and generates a device setup command in addition (the PostScript Level 2 operator **setpagedevice** as well as the DSC comment **%%DocumentMedia:** with *dpost*). The effect of this command can be the selection of a matching paper tray on a printer; on the other hand, the document may not print at all if no such tray is available. In general, it is recommended to use printer-specific options from a PPD file when the file is submitted to the print spooling system instead of this request. However, when generating PostScript as intermediate format with the intent of PDF creation, use of this request is recommended.
.cropat *L T W H*		ignored	T,**p**	Set the ''CropBox'' page parameter for PDF generation. The four arguments give the offset from the left (*L*) and top (*T*) margins of the document (as set by **mediasize**), and width (*W*) and height (*H*) of the box. The default units are points. The CropBox restricts the area of the page that is shown by a PDF viewer program. It is useful to hide cut marks and other printing instructions when the same PDF document is intended to be displayed on screen.
.trimat *L T W H*		ignored	T,**p**	Set the ''TrimBox'' page parameter for PDF generation and enables printing of marks when combined with *dpost –M*. The four arguments give the offset from the left (*L*) and top (*T*) margins of the document (as set by **mediasize**), and width (*W*) and height (*H*) of the box. The default units are points. The TrimBox specifies how the page is to be cut after it has been printed; it is sort of an electronic equivalent for cut marks (which should continue to be printed in addition).
.bleedat *L T W H*		ignored	T,**p**	Set the ''BleedBox'' page parameter for PDF generation. The four arguments give the offset from the left (*L*) and top (*T*) margins of the document (as set by **mediasize**), and width (*W*) and height (*H*) of the box. The default units are points. The BleedBox should be defined as a a frame around the objects of the actual document including any bleed areas (i.e. content that should extend to the end of the trimmed final page but is extended a bit such to work around possible cutting inaccuracies). Cut marks, color bars, and other information for the printing office should be positioned outside the BleedBox.
.bp ±*N*	*N*=1	–	B	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 effect the page number of the first page. The current page number is in the **%** register. The number of the next page is in the **.pn** register; this is either the value set by **pn** or the current page number plus 1.
.po ±*N*	0;1i	previous	**m**	Page offset. The current *left margin* is set to ±*N*. The *troff* initial value provides about 1 inch of paper margin on a typical device. The current page offset is available in the **.o** regis-

				ter.
.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**; this also works when the motion is downwards.

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 does not fit. An attempt is then made the hyphenate the word in effort to assemble a part of it into the output line. The spaces between the words on the output line are then increased to spread out the line to the current *line length* minus any current *indent*. A *word* is any string of characters delimited by the *space* character or the beginning/end of the input line. Any adjacent pair of words that must be kept together (neither split across output lines nor spread apart in the adjustment process) can be tied together by separating them with the *unpaddable space* character "\ " (backslash-space). The paddable no-break space character \~ keeps words on the same output line but may be spread during adjustment. The adjusted word spacings are uniform in *troff* and the minimum interword spacing can be controlled with the **ss** request (§2). In *nroff,* word spacings 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 current horizontal output position is in the **.k** register. The **.x** register indicates the remaining horizontal space on the current output line.

Text lines beginning with space characters and empty text lines (blank lines) cause a break. The filling of the line currently being collected is stopped and the line is output without adjustment. The **.blm** request can be used to redefine the processing of blank input lines. With the **.lsm** request the effect of lines beginning with spaces is changed. The read-only number register **lsn** contains the number of leading spaces. The read-only number register **lss** contains the horizontal space which corresponds to the leading spaces. Both registers are set also if **.lsm** is not used.

troff can optionally decrease word spacings, change the letter spacing, and reshape letters when adjusting lines to fit on both margins. The interword spacing controlled with the **ss** request is then understood as the optimum setting. Words are collected from input until the first word that would require to condense spacing or letter shapings is encountered. Between the end of the previous word and the end of this word, hyphenation points are additionally examined until the two possibilities to end the line surrounding the optimum are found. The point closest to the optimum is then chosen. If the word spacing so determined would fall below the minimum allowed, letter spacings and shapings are condensed. If it would exceed the threshold for expanded letter spacing and letter shaping, it is attempted to compensate until the threshold is reached, up to the maximum allowed letter

* Local motions have no effect on register **.h**.

spacing and letter shaping. Remaining space is distributed among the word spacings. Distribution among letter spacings and shaping is equal until one of the respective limits is reached. Lines that terminate with a *break* are normally set using the optimum word and line spacings unless the last word collected is the first word that falls below the optimum spacing, which may result in condensed spacing and shaping. The horizontal positions in the **.k** and **.x** registers are always computed using default character and spacing widths.

As an alternative to adjusting one line at a time as described, *troff* also supports adjusting one paragraph at once. In this mode, words are collected from input lines and requests are processed until a *break* occurs. The words are then split into lines such that the interword spaces are closest to the optimum setting across the entire paragraph. Ultimately, the lines so determined are printed. Traps, page breaks, and any positioning become effective only at this time. It is thus well possible that the current page number as read on the input line in the **%** register is lower than the number of the page on which the corresponding text is actually output. Any reference to the page number or page position (e.g. writing an index entry) should thus be handled using the output-line trap mechanism instead of in-line macros. Likewise the **.k** and **.x** registers are not meaningful in this mode, except that **.k** is never zero when text is present in a partial paragraph.

An input text line ending with **.**, **?**, **!**, or **:**, optionally followed by any number of **"**, **´**, **)**, **]**, *****, or **†**, is taken to be the end of a *sentence*, and an additional space character is automatically provided during filling. To prevent this at individual locations, add **\&** to the end of the input line; the second argument to the **ss** request (§2) changes the size of this space character and can also disable it globally. Both the sentence-ending and the transparent characters are configurable. The **\)** character is always transparent but behaves like **\&** in all other respects. Multiple inter-word space characters found in the input are retained, although the second character in a sequence of spaces following a sentence-ending character has the width of a sentence space; if the sentence space has been set to zero, any sequence of spaces following a sentence-ending character has the width of a single space. Trailing spaces are always discarded. Initial spaces are always retained and also cause a *break*.

With the **brnl** request, a *break* occurs at the end of each text input line in fill mode, except for lines interrupted with **\c**. Contrasting to nofill mode, text is still adjusted to the line length.

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.

When adjusting paragraphs at once, is possible to specify additional *penalties* for putting a line break after the current word (or part of a hyphenated word) by imbedding or attaching a **\j**´±*N*´ to it. A positive value of *N* discourages a line break, a negative value encourages it. Values of 1000000 and above are taken as infinitive penalties and always prevent a line break; values of –1000000 always cause a line break. A default penalty can be set with the **\J**´±*N*´ escape sequence; this is useful to discourage line breaks within a certain group of words, e.g. a person's name or a formula. Relative values in the argument to **\j** refer to the default penalty. The current default penalty is available in the **.defpenalty** number register. Separate penalties can be specified for breaking a line after a hyphenated word part using the **hypp** request (§13).

With the **brpnl** request, each end of a text input line in fill mode causes a *break* and a spread, except for lines interrupted with **\c**. The meaning of **\p** is then changed such that a line that it is attached to is *not* spread; this is the only method to achieve a regular *break* without a spread then. Manually adjusted text can thus be typed more comfortably since only the (fewer) lines that are not spread need to be marked.

A text input line that happens to begin with a control character can be made to not look like a control line by prefacing it with the non-printing, zero-width filler character **\&**. Still another way is to specify output translation of some convenient 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. The **.int** number register is set to a non-zero value if the previous line was interrupted and to zero otherwise. The length of a partial word is ignored for the value of the **.k** number register in *fill* mode; it is separately available with the **.kc** register.

4.3. Kerning. troff reads kerning tables from Type 1, OpenType, and TrueType font files. These tables contain small horizontal spacing adjustments for pairs of individual characters, e.g., the pair "Vo" would print as "Vo" without kerning applied. Placing **\&** between two characters disables kerning at that location; the **kern** request can disable it globally. The **kernpair** request adds a kerning pair; in contrast to predefined pairs, it allows the characters to originate from different fonts.

It is sometimes useful to add or substact a constant amount of spacing whenever a specific character appears; for example, «french» quotation marks usually require some additional distance to the words contained in them. The **kernafter** and **kernbefore** requests allow to define such adjustments. Again, a **\.** disables them at individual points; they are generally not applied if the other character is a space.

4.4. Hanging characters. Characters can hang beyond the left or right margins of adjusted text; the **lhang** and **rhang** requests specify this. Left margin adjustments are evaluated before the letters that fit on the current line are computed, and can thus principally be of any length. In contrast, right margin adjustments are evaluated after this computation is finished, and the adjustment is simply added to the word space of the output line. Thus a positive right adjustment that is large in relation to the line length will cause visible holes, and a negative adjustment will ultimately cause the words on the line to be printed over each other. This is not a problem for the typical application of hanging punctuation for visual alignment, though; if e.g. a line with eight word spaces is shifted by .08 em, each word space is enlarged by only .01 em.

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 (This can be changed with the **.blm** and **.lsm** requests.).
.lsm *xx*	none	none	–	Whenever a line beginning with spaces is encountered, the macro *xx* is invoked instead of the default behavior to cause a break. If the spaces are followed by an escape, that may be processed before the macro is called (e.g. font changes). To avoid this prepend that escape with **\&**.
.brp	–	–	B	Break and spread; same effect as the **\p** escape sequence except that it causes the remaining part of a paragraph shape as set with **pshape** to be discarded.
.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.

Indicator	Adjust Type
l	adjust left margin only
r	adjust right margin only
c	center
b or **n**	adjust both margins
p or **pb**	adjust both margins paragraph-wise
pl	adjust left margin paragraph-wise
pr	adjust right margin paragraph-wise
pc	center paragraph-wise
absent	unchanged

The number register **.j** contains the current value of the **ad** setting; its value can be recorded and used subsequently to set adjustment.

.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.
.padj *N*	off	on	–	Control paragraph-at-once adjustment globally. If *N*≠0 or missing, paragraph-at-once adjustment is enabled in all environments, and **ad** *x* effectively acts like **ad p***x*. The current value is available in the **.padj** register.
.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. The number of lines to be right-aligned, if any, is set to zero. The remaining number of lines to be centered is available in the **.ce** register.
.rj *N*	off	*N*=1	B,E	Right-align the next *N* input text lines within the current (line-length minus indent); otherwise like **ce**. The number of lines to be centered, if any, is set to zero. The remaining number of lines to be right-justified is available in the **.rj** register.
.brnl *N*	off	*N*=∞	B,E	Break at end of next *N* input text lines when filling is in effect. The remaining number of lines so treated is available in the **.brnl** register.
.brpnl *N*	off	*N*=∞	B,E	Break and spread at end of next *N* input text lines when filling is in effect. The remaining number of lines so treated is available in the **.brpnl** register. **brpnl** disables **brnl** and vice-versa.
.minss *N*	off	off	E,T	Minimum word space. When adjusting both margins, *troff* may decrease the size of the word space down to *N*/36 ems (rather than to the value set by **ss**). The current value is available in the **.minss** register.
.letadj *X U S Y V*		off	E,T	Dynamic letter spacing and reshaping when adjusting lines. The space between letters can be automatically expanded or condensed (in addition to the space between words), and the width of letters can be automatically changed in **ad b** mode. The *X* argument gives the minimum percentage of an en of the current point size by which adjacent characters may be tightened; the *U* argument gives the minimum percentage of the letter width by which letters may be condensed. When the computed size of the word space would have to be larger

				than *S*, additional space will be inserted between letters up to a limit of *Y* percent of an en of the current point size, and letters will be expanded by up to *V* percent of their width until the word space does not extend beyond *S* anymore. *S* is given in units of 1/36 ems as with **ss**. The number registers **.lspmin**, **.lshmin**, **.letss**, **.lspmax**, and **.lshmax** contain the current values of *X*, *U*, *S*, *Y*, and *V*, respectively.
.sentchar *c...*	**.?!:**	off	E	Sentence-ending characters. When one of the characters *c...* appears at the end of an input text line, an additional space character of the size defined with the **ss** request is inserted. The current set of sentence-ending characters is available in the **.sentchar** number register.
.transchar *c..*	**"')]*†**	off	E	Transparent characters for sentence-ending. A sentence-ending character is recognized as such even if followed by one or more of the characters *c...* before the end of an input text line. The current set of transparent characters for sentence ending is available in the **.transchar** number register.
.track *F S N T M*		ignored	P,T,**p**	Static letter space tracking. If the current font is *F* and the point size is below or equal to *S*, white space of width *N* is added to each character. If the point size is above or equal to *T*, white space of width *M* is added. If the point size is between *S* and *T*, the amount of white space added is computed as a value between *N* and *M* using the current point size *s*: $(sM{-}sN{+}TN{-}SM)\,/\,(T{-}S)$. The default unit for all numeric arguments is points. Negative numbers are accepted and cause a decrease of letter space. No adjustment is performed on the last character of an output line. Tracking also applies to characters from another font that have been selected by the **fallback** sequence.
.kern *N*	1	1	P,T	Control pairwise kerning; disabled if *N*=0, otherwise enabled.
.fkern *F N*	1	1	P,T	Control the use of kerning tables from font *F*; disabled if *N*=0, enabled if *N*=1 or missing. For *N*≥2, only kerning pairs with absolute values greater or equal to *N* are used. *troff* kerning adjustments as defined by the following requests are not affected.
.kernpair *F c... G d... N*		ignored	P,T	Add a kerning pair to the kerning table for character *c* from font *F* and character *d* from font *G*. *c* and *d* may consist of multiple characters; in this case, table entries are added for any pair combination of characters from *c* and *d*. The *N* argument is 1/72 000 of an inch multiplied by the current point size or 1/1000 of an em (as in AFM kerning pair definitions); it may be negative. To add a kerning pair that includes the space character, use “\ ”.
.kernafter *F c... N d... M ...*			P,T	Add a constant amount of space after a character if the current font is *F* and *c* is the first character of a pair of characters subject to kerning. *c* may consist of multiple characters; in this case, the same amount is added whenever one of the given characters appears. The *N* argument is 1/72 000 of an inch multiplied by the current point size or 1/1000 of an em (as in AFM kerning pair definitions); it may be negative. Same for *d/M* etc.

.kernbefore *F c... N d... M ...*		P,T	Add a constant amount of space before a character if the current font is *F* and *c* is the second character of a pair of characters subject to kerning. *c* may consist of multiple characters; in this case, the same amount is added whenever one of the given characters appears. The *N* argument is 1/72 000 of an inch multiplied by the current point size or 1/1000 of an em (as in AFM kerning pair definitions); it may be negative. Same for *d/M* etc.
.lhang *F c... N d... M ...*	ignored	T	Hanging characters at left margin. When the current font is *F* and *c* appears at the left margin of an output line in left-adjusted, both-adjusted, or nofill mode, the margin is relocated to the right by *N*, which is 1/72 000 of an inch multiplied by the current point size or 1/1000 of an em (as in AFM character width definitions); it may be negative. *c* may consist of multiple characters; in this case, the margin is relocated whenever one of the given characters appears. Same for *d/M* etc.
.rhang *F c... N d... M ...*	ignored	T	Hanging characters at right margin. When the current font is *F* and *c* appears at the right margin of an output line in right-adjusted, both-adjusted, or nofill mode, the margin is relocated to the right by *N*, which is 1/72 000 of an inch multiplied by the current point size or 1/1000 of an em (as in AFM character width definitions); it may be negative. *c* may consist of multiple characters; in this case, the margin is relocated whenever one of the given characters appears. Same for *d/M* etc.

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. *V* should 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**, but it is better to use a large **vs** instead; certain preprocessors assume single spacing. The current line spacing is available in the **.L** register.

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. 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.

In **\x**´...´ and other functions having a pair of delimiters around their parameter (here ´), the delimiter choice is arbitrary, except that it can not look like the continuation of a number expression for *N*.

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**.

5.4. Ascenders and descenders. Type 1, OpenType, and TrueType metrics supply information about the typical extents of characters above the base-line (ascender) and below it (descender). Usually, these correspond to the top of the lowercase ‘‘d’’ and the bottom of the lowercase ‘‘p’’, respectively. The **.ascender** and **.descender** number registers contain these values in units with correct scaling for the current point size applied. If no values are available, these registers are set to zero.

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. 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. (Recall the use of **.sp** |*N* from §1.3.) 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**. The number register **.ns** is set to a non-zero value in no-space mode and to zero otherwise.
.rs	space	–	D	Restore spacing. The no-space mode is turned off.
Blank text line.		–	B	Causes a break and output of a blank line exactly like **sp 1** unless a different action has been specified with the **blm** request.

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 **.y** and **.in** registers both hold the indent that actually applies to the current line, taking a temporary indent into account. The length of *three-part titles* produced by **tl** (see §14) is *independently* set by **lt**.

In **ad p** mode, indent, temporary indent, and line length should be predefined for the entire paragraph at the time a *break* occurs. To achieve this with indent and line length varying inside a paragraph, a shape can be defined with **pshape**. The **ll** and **in** requests are also effective while formatting a paragraph but may result in less optimal line breaking decisions then.

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
.ll ±*N*	6.5 in	previous	E,**m**	Line length is set to ±*N*.
.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.
.pshape ±*I1* ±*L1* ±*I2* ±*L2* ... off			E,**m**	Set a special shape for the current paragraph in **ad p** mode. At the next *break* when the paragraph gets formatted, its first line is indented by *I1* and has length *L1*, its second line is indented by *I2* and has length *L2*, and so forth. Relative numbers refer to the previous indent or line length pair, or to the values set by **in** and **ll** for the first pair. The last of the indent and line length pairs stays effective if the paragraph has more lines than pairs are given. With an odd number of arguments, the standard line length as set by **ll** is used at the end. Once the current paragraph has been printed, the shape specification is forgotten, and the standard indent and line length values are in effect again unless another **pshape** request occurs. As an exception, **\p** preserves the remaining part of the shape. To produce shapes with holes as in this example, diversion traps can be used to move formatted lines in vertical direction.

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 consist of an arbitrary number of ASCII characters (§1.1) and may usurp previously defined request, macro, or string names; this implies that build-in operators may be (irrevocably) redefined. 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 arbitrarily many *arguments*. The strings *x*, *xx*, and *xxx* are interpolated at any desired point with *x, *(xx, and *[xxx] respectively; the form *[xxx arg ...] allows to specify arguments to a string. 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.
- Environment variables indicated by **\V** are interpolated.
- Concealed newlines indicated by \(newline) are eliminated.
- Comments indicated by **\"** or **\#** 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 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 character. The argument **""** is explicitly null. If the desired arguments won't fit on a line, a concealed newline may be used to continue on the next line. A trailing double quote may be omitted.

Similarly, arguments may be passed to strings with the ***[** syntax, separated by spaces, until the argument list ends with the **]** character. String arguments are otherwise handled exactly like macro arguments.

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*, **\$(***nn*, or **\$[***nnn***]**, which interpolates the *n*th, *nn*th, or *nnn*th argument, respectively. 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 escape sequence **\$*** interpolates all arguments to a macro, separated by spaces; **\$@** interpolates all arguments, each one surrounded by double quotes, separated by spaces. The name of the current macro or string is available with the **\$0** escape sequence. The number of currently available arguments is in the **.$** register.

No arguments are available at the top (non-macro) level or within a trap-invoked macro.

Arguments are copied in *copy mode* onto a stack where they are available for reference. 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). The current diversion level is available in the **.dilev** register.

A previous partially filled line is included when a diversion begins. A partially filled line at the end of a diversion is not included but becomes part of the surrounding diversion level. With the **box** request, a previous partially filled line is not included. At the end of the diversion, this partially filled line is restored at the surrounding level, discarding any partially filled line from within the diversion. The behavior is otherwise the same as with a standard diversion.

7.5. Traps. Five types of trap mechanisms are available—page traps, output-line traps, a single diversion trap, multiple diversion traps, 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.

An output-line trap is planted using the escape sequence **\P***x*, **\P(***xx*, or **\P[***xxx***]**. It is converted to an internal character code that becomes part of the current text. When the line containing it has been physically output, the macro *x*, *xx*, or *xxx*, respectively, is invoked. A line may contain more than one output-line trap; they are invoked in the order they appear. If a page trap becomes effective at the same point, it is invoked immediately after all output-line traps. Output-line traps are not invoked at the time they become part of a diversion. They can thus be used to determine e.g. the effective page number of the surrounding text, even when it had originally been diverted on another page.

A single macro-invocation trap effective in the current diversion may be planted using **dt**. If another **dt** follows in the same diversion, the trap position is changed. Multiple traps may be planted in the current diversion using **dwh** and **dch**. The **.t** register works in a diversion; if there is no subsequent trap a *large* distance is returned. For a description of input-line-count traps, see **it** below.

7.6. Recursion. troff macros can be invoked recursively. In general, return information and arguments for macro calls are stored as *frames* on a *stack*. Since the size of the stack is limited by available memory, recursive calls cannot descend to arbitrary depths. To avoid consumption of large amounts of memory in case of endless loops, the depth is deliberately restricted further; the **recursionlimit** request allows to adjust that restriction.

A macro call can be *tail-recursive*, that is, the re-invocation of the current macro can occur in its last statement. *troff* can then replace the stack frame of the current macro instance with that of the following one, and the stack size remains constant. The depth of tail-recursive macro calls is thus principally unlimited, and is not artificially restricted by default. Since *troff* cannot determine the control character at the time the macro is executed in advance, the recursive macro call must not only be the last statement of a macro, but must actually occur in the last line for tail-recursion elimination to become effective, although it may be prefixed by **.if** or **.el** or followed by **\}**.

7.7. Local strings. Strings, macros, and diversions are, once defined, normally accessible from any part of a *troff* program. Local strings, defined with **lds**, are accessible only within the currently executing macro instance, and are not inherited to macros invoked from them. When the currently executing macro terminates, they are automatically deleted, and all associated storage is reclaimed.

As long as a local string named *xx* exists, any reference with **\(***xx* or **\[***xx***]** retrieves the value of the local string; the global string *xx*, if any, becomes inaccessible by this method. Calls to **as**, **substring**, **index**, and **chop** modify the local string. **watch** with the name of a local string as argument notifies on modifications of the local string. **rm** removes a local string and possibly makes a global string visible again, **rn** renames a local string to another local string, and **als** creates a local alias to a local string.

It is not allowed to define local macros or diversions. Calls to *.xx* or *´xx* reference a global macro or diversion even if a local string *xx* exists. Traps and the **\Y** escape sequence always operate on global macros or diversions.

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**).

.lds *xx string* –	ignored	–	Define local string *xx* containing *string*. Equivalent to **ds** at the top non-macro level.
.substring *xx N* [*M*]	*M*=−1	–	Replace string *xx* by its substring between indices *N* and *M*. *N* and *M* start at 0. Negative values are interpreted relative to the end of the string; −1 implies the last character of the string.
.length *R string*	*R* set to 0	–	Store the length of *string* in register *R*. *string* is read in *copy mode.*
.index *R xx string*	ignored	–	Store the position of the first occurence of *string* in *xx* in register *R*. Positions are counted from 0. If no occurence is found, *R* is set to –1. *string* is read in *copy mode.*
.chop *xx* –	ignored	–	Remove the last character of the macro, string, or diversion *xx*.
.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. If many macros and strings are being created dynamically, it may become necessary to remove unused ones to recapture internal storage space for newer registers. If a macro is removed while it is in use, associated storage is not released, and a warning of the **mac** category is emitted. **rm** may have an unlimited number of arguments. However for creating portable documents only one argument to **rm** should be used.
.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**).
.box *xx* –	end	D	Divert output to macro *xx*, excluding a partially filled line.
.boxa *xx* –	end	D	Divert and append to *xx*, excluding a partially filled line.
.unformat *xx* –	ignored	–	Strip line break information from diversion *xx*. All breaks that do not result in explicit vertical movement are discarded, interword spaces that had been converted to horizontal movements during adjustment become space characters again, the effects of tabulators and field characters are reverted, and hyphenated word parts are combined again.
.asciify *xx* –	ignored	–	All characters in diversion *xx* changed to plain text. Has all effects described for **unformat** and additionally discards font and point size information and splits substituted ligatures to their individual characters.
.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.
.dwh *N xx*	–	–	D,**v**	Set location trap in current diversion. Diversion traps planted with **dt** are not affected; if both a **dwh** and a **dt** trap are set at the same position, both are effective. A **dwh** trap previously set at the same position is replaced. At the top non-diversion level, it is equivalent to **wh**. Otherwise with a negative or zero *N*, the trap will never be invoked since a diversion has no bottom and can never reach its top again.
.dch *xx N*	–	–	D,**v**	Change trap location in current diversion; equivalent to **ch** at the top non-diversion level.
.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.
.vpt *N*	1	ignored	–	Enable ($N \neq 0$) or disable (N=0) page ejections and vertical position traps, i.e. those set with **wh** or **dt**. At end of input, the page is forcefully ejected. The number register **.vpt** holds the current setting.
.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.
.itc *N xx*	–	off	E	Set an input-line-count trap like **it**, but ignore line interruptions with **\c** when counting lines.
.return	–	–	–	Immediately return from the current macro to the level above.
.shift *N*	–	1	–	Shift the arguments to the current macro by *N*.
.als *yy xx*	–	–	–	*yy* is created as an alias for the request, macro, or string *xx*. The alias name is in every respect identical to the original name. If *xx* is removed or renamed, *yy* continues to refer to the object at the time the alias had been created (and vice-versa). If either *xx* or *yy* are redefined, both refer to the new definition.
.blm *xx*	none	none	–	Whenever a blank line is encountered, the macro *xx* is invoked instead of the default behavior to output a blank line.
.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, but all processing ceases at the next page eject.
.recursionlimit *N M*		–	–	Set the maximum stack depth for generally recursive invocations of macros to *N*, and for invocations of tail-recursive macros to *M*. If *N* or *M* are zero, the respective depth is unlimited. The default is 512 for the general case, and no limit for tail-recursion.

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[*xxx***]**	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*
\n+[*xxx***]**	*xxx* incremented by *M*	*N*+*M*
\n–[*xxx***]**	*xxx* decremented by *M*	*N*–*M*

Floating-point registers can be created and modified using **nrf**. They share a common name space with integer registers; a **nr** request turns a floating-point register into an integer register. Access and auto-increment are as with integer registers.

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**.

Local number registers can be created and modified using **lnr** and **lnrf**. Like local strings, local registers are accessible only within the currently executing macro instance, are not inherited to macros invoked from them, and are deleted when the current macro returns.

The existence of a local number register *xx* makes the global number register *xx*, if any, completely inaccessible. The sequences **\n(***xx* and **\n[***xx***]** retrieve the value of the local register, **\R***'xx…'*, **\n+(***xx* and **\n+[***xx***]** (and likewise) modify it, and **\g(***xx* and **\g[***xx***]** retrieve its format. Subsequent calls to **nr**, **nrf**, **af**, **length**, and **index** with *xx* as argument modify the local register. **watchn** notifies on modifications of the local register. **rr** removes a local register and possibly makes a global register visible again, **rnn** renames a local register to another local register, and **aln** creates a local alias to a local register.

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*. An alternate syntax is available with the **\R***´R* ±*N´* escape sequence.
.nrf *R* ±*F G*		–	**u**	The floating-point register *R* is assigned the value ±*F* with respect to the previous value, if any. Comparison computations in the expression *F* are performed using floating-point values. The increment for auto-incrementing, which also is a floating-point value, is set to *G*.
.lnr *R* ±*N M*		–	**u**	Define and set local number register *R*.
.lnrf *R* ±*F G*		–	**u**	Define and set local floating-point register *R*.
.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, the *width* function (§11.2), and floating-point registers are always arabic. Warning: the value of a number register in a non-Arabic format is not numeric, and will not produce the expected results in expressions.

The function **\g***x*, **\g(***xx*, or **\g[***xxx***]** returns the format of a number register in a form suitable for **af**; it returns nothing if the register has not been used.

.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. The register **.R** contains the number of number registers still available.

rr may have an unlimited number of arguments. However for creating portable documents only one argument to **rr** should be used.

.rnn *R S* – – – Rename register *R* to *S*. If *S* exists, it is first removed.

.aln *S R* – – – Register *S* is created as an alias for *R*. The alias name is in every respect identical to the original name. If *R* is removed or renamed, *S* continues to refer to the register at the time the alias had been created (and vice-versa). A change in value or format in *R* affects *S* and vice-versa.

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* ...	8 n; 0.5 i	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 0.8in. The stop values are separated by spaces, and a value preceded by **+** is treated as an increment to the previous stop value. The **.S** register* and the **.tabs** register hold a string with the current tab stops in a form that is acceptable for the **ta** request; it can thus be used to save and restore tab stops.
.ta *Mv* ... *Nw* **T** *At* ... *Zu*a	*N*=0		E,**m**	Set repeated tab stops and types at *Mv*, ..., *Nw*, *N* + *At*, ..., *N* + *Zu*, *N* + *Z* + *At*, ..., *N* + *Z* + *Zu*, *N* + 2·*Z* + *At*, ..., *N* + 2·*Z* + *Zu*, *N* + 3·*Z* + *At*, ..., *N* + 3·*Z* + *Zu*, ... Thus **.ta T 0.5i** sets tab stops every 0.5in, and **.ta 1i 4i T .25i 1i** sets tab stops at 1in, 4in, 4.25in, 5in, 5.25in, 6in, 6.25in, etc.
.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 e_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. \ should not be confused with the ASCII control character ESC of the same name. The escape character \ can be output by using the special character **\(rs**. 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; this escape sequence is interpreted in *copy mode*. **\E** evaluates to the current escape character but is not interpreted in *copy mode*; if it is put into a string or macro, it will introduce an escape sequence once the string is printed or the macro is executed, respectively. **ecs** and **ecr** save and restore the escape character. If necessary or convenient, the escape mechanism may be turned off with **eo**, and restored with **ec**.

* Register **.S** is available for compatibility with DWB.

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.
.ecs	\\	–	–	Save escape character.
.ecr	\\	–	–	Restore saved escape character.
.escoff *esc* ...	on	ignored	–	Disable escapes *esc* ... . For example the request **.escoff ; @** would disable escapes **\\;**, \\@{, and \\@}. The request had been added as a workaround for manual pages which by mistake escape normal charaters. At the moment only one-character arguments are valid. This may change in future versions.
.escon *esc* ...	on	ignored	–	Enable escapes *esc* ... .

10.2. Ligatures. The set of available ligatures is device and font dependent, but is often a subset of **fi**, **fl**, , , and . They may be input (even in *nroff*) by **\\(fi**, **\\(fl**, **\\(ff**, **\\(Fi**, and **\\(Fl** respectively. In *troff,* the **flig** request specifies the set of ligatures available with an individual font. The ligature mode is normally on in *troff,* and *automatically* invokes ligatures during input. At most the five named ligatures are enabled by default.

The ligature suppressor **\\;** disables automatic ligature building if it is placed between two characters, but has no other effects. Many other escape sequences, e.g. **\\&**, **\\%**, also disable automatic ligature building.

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.*
.flig *F string c* ...		ignored	T	Define the set of ligatures to be used with font *F*. *string* may consist of up to four characters; its exact occurences are substituted by the character *c* whenever the current font is *F*. If *string* starts with a minus sign, ligature subtitution for it is disabled; the *c* argument must not be present in this case. It is required that any substring of *string* that consists of two or more characters is also defined as a ligature. Multiple *string/c* pairs may be given. As a special case, **.flig** *F* **0** disables all ligatures for font *F*. The default set of ligatures is specific to a font and is determined from its metrics file. As examples, **.flig R Th \\[T_h]** enables a ligature for "Th" in font **R**, and **.flig R –Th** disables it.
.fdeferlig *F string* ...		ignored	T	Defer ligature building for the first character of *string*. Normally when a sequence of three characters appears for which both the first and the last two characters are defined as a ligature, the ligature is built using the first two characters. For example, if "ffi" appears in input and both "ff" and "fi" are defined as ligatures, the "ff" ligature is built, followed by a single "i" character. Use of this request causes the ligature to be built using the second two characters, so in the example, **.fdeferlig R ffi** would result in a single "f" character followed by a "fi" ligature.

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 that on font position 2 (normally Times Italic, 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	Continuous underline. 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.

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). Text in a diversion is not translated again when it is output or rediverted unless the **asciify** request had been applied on it.

A character can be substituted by an arbitrary sequence of characters, motions, drawing commands, etc. by means of the **char** request.

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.
.trin *abcd....*	none	–	O	Translate *a* into *b*, *c* into *d*, etc. as with **tr**, but when the **asciify** request is called, retranslate *b* to *a*, *d* to *c*, etc. in the diversion it is applied to.
.trnt *abcd....*	none	–	O	Translate *a* into *b*, *c* into *d*, etc. as with **tr**, unless the output is written to a diversion in transparent mode.
.ftr *F abcd....*	none	–	P,T	Font-specific **tr**. If the current character originates from font *F*, translate character *a* into *b*, *c* into *d*, etc.
.char *c string*		ignored	–	Define character *c* to *string*. *string* is read in *copy mode* when the request is processed. Whenever *c* occurs in regular (not copy mode) later, a temporary copy of the current environment is created, the escape

character is temporarily set to \, and *string* is output in nofill mode to an unnamed diversion. If *c* itself occurs in *string*, no recursive substitution takes place, but the plain character *c* is selected instead. The resulting object continues to behave like the single character *c* in the data stream, notionally retaining its font and character size properties. It can thus be part of a kerning definition, can be hyphenated (possibly requiring an explicit hyphenation code to be assigned using the **hcode** request), can become the tab or leader character, and can be used for line drawing.

Static letter space adjustments defined with the **track** request are applied to each but the last character inside the diversion, and to the object as a whole. Dynamic letter space adjustments defined with the **letadj** request are only applied to the object as a whole. No letter reshaping is performed on either the object or the contents of the diversion.

If both **tr** and **char** are defined for a character *c*, **tr** becomes effective first and changes *c* to the translation defined, but if *c* is the result of **tr** and **char** is defined for it, **char** is applied. **ftr** has no effect on characters for which **char** is defined. Both **tr** and **ftr** are applied to characters inside the **char** diversion as usual.

At the time *c* would have been output, the contents of the diversion are printed instead of it.

.fchar *c string*		ignored	T	Define character *c* to *string* as a fallback only: If *c* is present in the current font, output *c*, otherwise, output *string* as with **char**.
.rchar *c...*		ignored	–	Remove character definitions for *c...*; applies to both **char** and **fchar**.
.utf8conv *c1 c2*		ignored	–	*nroff* only: With **–Tlocale** and a UTF-8 locale, "**-**" is replaced by a UTF-8 hyphen, and "**\-**" remains the ASCII hyphen-minus character. This is because in manual pages, "**\-**" represents the ASCII option introduction character, and converting it to a UTF-8 minus character would make it impossible to copy-and-paste option descriptions. Similar considerations apply to ` ' vs. \` \'. The former are typographic single quotes, while the latter are commonly used for the ASCII syntax quotes in manual pages. By use of **utf8conv** the three default translations can be redefined and additional translations can be done. If *c2* is missing a translation is removed. This is applied for manual pages where the no translation is used.

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.

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
.output *string*		ignored	–	Write *string* directly to intermediate output, regardless of whether there is a current diversion or not. *string* is read in copy mode, and an initial double-quote is discarded.

10.7. Transparent output. The sequence **\X**´*anything*´ copies *anything* to the *troff* output, as a device control function in the form **x X** *anything* (§26). Escape sequences in *anything* are processed. The sequence **\Y***x*, **\Y(***xx*, or **\Y[***xxx***]** copies the contents of the string or macro *x*, *xx*, or *xxx*,, respectively, to the output as a device control function without processing escape sequences. Newlines in the macro are embedded in the output. *nroff* discards transparent output sequences and their contents.

10.8. 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 **\"**. This form does not conceal the newline at the end of the comment. 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 **.\"**. The form **\#** includes the newline as part of the comment. It thus effectively acts like a concealed newline in concatenating the following line immediately to the current line.

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 rowspan="2">\h´N´
\(space)
\~
\0</td><td colspan="2" rowspan="2">Move distance N
Unpaddable space-size space
Paddable no-break space
Digit-size space</td></tr>
<tr><td rowspan="2">\u
\d
\r</td><td rowspan="2">½ em up
½ em down
1 em up</td><td rowspan="2">½ line up
½ line down
1 line up</td></tr>
<tr><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**; note 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´\fB1. ´u** could be used to temporarily indent leftward a distance equal to the size of the string "**1.** " in font **B**.

The width function also sets five 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**. The registers **rst** and **rsb** are set respectively to the highest and lowest visual extent of *string* relative to the baseline, i.e. to the maximum and minimum extent of the *y* value of any bounding box of the characters in *string*. 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.

The **.w** number register contains the width of the previous character independently of the width function. Similarly, the **.cht** and **.cdp** registers are set respectively to the highest and lowest visual extent of the previous character relative to the baseline.

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+3u´***word* will embolden *word* by

* The line drawing escapes \l and \L also cause local motions.

backing up to almost its beginning and overprinting it, resulting in *word*. Likewise, **\k(***xx* and **\k[***xxx***]** will store the horizontal position in register *xx* or *xxx*, respectively. The **hp** number register also holds the current horizontal position in the input line.

12. Overstrike, Bracket, Line-drawing, Graphics, 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* overprinted with centers aligned; the total width is that of the widest character. *string* may *not* contain local vertical motion. As examples, **\o´e\´´** produces **é**, and **\o´\(mo\(sl´** produces ∉.

12.2. Zero-width characters and strings. 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 |‾_|.

The function **\Z***'string'* prints *string* in nofill mode and restores the horizontal and vertical position afterwards.

12.3. Large Brackets. The Special 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 $\begin{bmatrix}\mathbf{E}\end{bmatrix}$.

12.4. Line drawing. The function **\l´***Nc***´** (backslash-ell) will draw a string of repeated *c*'s towards the right for a distance *N*. 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. If *N* is less than the width of *c*, a single *c* is centered on a distance *N*. 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; the set of these characters can be customized with the **connectchar** request described below. 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***´** draws 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. Motions of line drawing functions are local which means that the effect of these motions vanishes when a new output line is started.

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.

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
.connectchar *c*..	**"\(ru\(ul\(rn**	off	E	Connected characters for line drawing. If there is a remainder to be drawn for a horizontal line, these characters are drawn over-lapping. The current set is available in the **.connectchar** number register.

12.5. Graphics. The function **\D**´*c...*´ draws a graphic object of type *c* according to a sequence of parameters, which are generally pairs of numbers.

\D´l *dh dv´*	draw line from current position by *dh, dv*
\D´p *dh1 dv1 dh2 dv2 ...´*	draw polygon, i.e. a line to *dh1, dv1*, then to *dh2, dv2*, then ...
\D´P *dh1 dv1 dh2 dv2 ...´*	draw filled polygon
\D´c *d´*	draw circle of diameter *d* with left side at current position
\D´C *d´*	draw filled circle
\D´e *u v´*	draw ellipse of diameters *u* and *v*
\D´E *u v´*	draw filled ellipse
\D´a *a b c d´*	draw arc from current position to *a*+*c*, *b*+*d*, with center at *a*, *b* from current position
\D´~ *a b c d...´*	draw B-spline from current position by *a*, *b*, then by *c*, *d*, then by *c*, *d*, then ...

For example,**\D´e0.2i 0.1i´** draws the ellipse , and **\D´l.2i –.1i´\D´l.1i .1i´** the line . A **\D** with an unknown *c* is processed and copied through to the output for unspecified interpretation; coordinates are interpreted alternately as horizontal and vertical values.

Numbers taken as horizontal (first, third, etc.) have default scaling of **m**; vertical numbers (second, fourth, etc.) have default scaling of **v** (§1.3). The position after a graphical object has been drawn is at its end; for circles and ellipses, the "end" is at the right side.

13. Hyphenation.

Automatic hyphenation may be switched off and on. When switched on with **hy**, several variants may be set. A *hyphenation indicator* character, by default **\%**, may be imbedded in a word to specify desired hyphenation points, or may be prefixed to suppress hyphenation. The character **\:** indicates that a line break may optionally occur at a point, but that no hyphen is to be generated. In addition, the user may specify a small list of exception words.

Only words that consist of a central alphabetic string surrounded by (usually null) non-alphabetic strings are considered candidates for automatic hyphenation. Unless the set of optional line break characters is otherwise configured, words that contain hyphens (minus), em-dashes (**\(em**), or hyphenation indicator characters are *always* subject to splitting after those characters, whether 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*≥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. For *N*= 16 and 32, the last and first characters respectively of a word are allowed to be split off; this is only effective for explicit hyphenation points specified with **\%**, **\:**, or **hw**. These values are additive; i.e. *N*= 14 will invoke the three restrictions. The current value is available in the **.hy** number register.
.hylang *name*	off	off	E	Set the hyphenation language to *name*, which is one of **de_DE**, **de_DE@traditional**, **en_US**, **fr_FR**, **it_IT**, **la_VA**, or **nl_NL**. Other languages can be made available by adding hyphenation files to the directory **/usr/ucblib/doctools/hyphen**. If no *name* argument is present, the hyphenation is reset to the traditional *troff* mechanism. The current value is available in the **.hylang** number register. This request also makes parts of words composed by **-** or **\(em** characters subject to hyphenation, unless otherwise defined explicitly with the **.nhychar** request. Traditionally, the only hyphenation points in such words had been the hyphens/dashes.
.shc *c*	-	-	E	Set the soft hyphenation character, i.e. the character that is inserted at the end of a hyphenated word. The current soft hyphenation character is available in the **.shc** number register.
.hcode *abcd...*		–	E	Hyphenation code. When determining the hyphenation points for an input word, *a* is mapped to *b*, etc. When no hyphenation code is specified for a character, it is mapped to its lowercase variant if necessary, and the "long s" (**\[longs]**) character is mapped to a "round s".
.hylen *N*	5	5	E	Hyphenate only words of at least *N* characters in length. The current value is available in the **.hylen** number register.
.hlm *N*	off	off	E	Maximum number of consecutive hyphenated lines. Each time a line is hyphenated automatically, the count of consecutive hyphenated lines (accessible in the **.hlc** number register) is incremented; whenever a line is not automatically hyphenated, it is reset to zero. This request allows to set a limit on the maximum number of consecutive hyphenated lines; when the count of consecutive lines has reached the maximum, the current line is not hyphenated. The default is no limit. The current value is accessible in the **.hlm** number register.
.hypp *N M L*	0 0 0	0 0 0	E	Define hyphenation penalties for **ad p** mode. Every inserted hyphen is given a penalty of *N* when computing optimal break points; each hyphen that is followed by another hyphen gets an additional penalty of *M*. A hyphen that is inserted in the last word of a paragraph gets an additional penalty of *L*. A value of zero means no penalty. Effective penalties correspond to values between 10 and 200. The current values are available in the **.hypp**, **.hypp2**, and **.hypp3** number registers.
.breakchar *c.*	**-—**	off	E	Optional line break characters. A line may always be split after one of these characters. The current set of optional line break characters is available in the **.breakchar** number register.

.nhychar *c...*	**-—**	off	E	Hyphenation-inhibiting characters. A word that contains one of the characters *c...* is not hyphenated, except that it may be split across lines if one of these characters is also an optional line break character. The current set of hyphenation-inhibiting characters is available in the **.nhychar** number register.
.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.

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. The current value is available in the **.lt** number register.

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
3 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
6 **.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.

9 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
12 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 an integer numerical expression, *F* is a floating-point 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 $N \leq 0$ [sic], accept *anything*.
.if f*F anything*		–	**u**	If floating-point expression $F > 0$, accept *anything*.
.if !f*F anything*		–	**u**	If floating-point expression $F \leq 0$ [sic], 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 of the forms for **if** above are valid.
.el *anything*		–	–	Else portion of if-else.
.while *c anything*		–	–	Execute *anything* while *c* is true; all of the forms for **if** above are valid. *anything* is stored in an unnamed temporary macro each time a **while** loop is prepared for execution. In the multi-line case, the **\\}** terminating the loop must be placed at the end of a line. When *anything* is copied to the macro, neither copy mode nor regular escape interpretation apply. References to number registers, strings, etc. must thus be written using a single escape character unless the **while** request is contained in a macro definition. Nested loops will result in many macro definitions and may slow down execution, especially if their body is large. Recursive macros (§10.6) do not suffer from this problem. In contrast to recursive macro calls, there is no mechanism to terminate a loop automatically when a repetition limit has been reached. In case of doubt, an explicit limit termination condition should be added to prevent endless loops.
.break *n*	–	1	–	Break out of *n* nested **while** loops, or terminate the current loop if no argument is given. It is not necessary that all of the loops are contained within the same macro; if there are any macros executing inside the specified loop, control also returns from these macros. In case of a non-positive or non-numeric argument, *n*=1 is assumed. If the number of levels requested is greater than the number of loops currently executing, con-

trol returns to the highest non-looping level.

.continue *n*	–	1	–	Continue at the *n*-th nested **while** loop, or continue the current loop if no argument is given. Execution resumes with the test of the specified **while** loop; if this test fails, the request is effectively like **break**. **continue** also returns from all inside macro calls until it has reached the specified loop. In case of a non-positive or non-numeric argument, *n*=1 is assumed. If the number of levels requested is greater than the number of loops currently executing, control returns to the highest non-looping level, and no loop is continued.

The built-in condition names are:

Condition Name	True if
c *G*	Character *G* exists in the current font, where *G* is either an ASCII or localized input character, a *troff* special character **\(***xx* or **\[***xxx***]**, or a **\U**´*X*´ escape sequence
d *xx*	There is a request, macro, or string *xx*
r *xx*	Number register *xx* has been accessed
F *font*	Font named *font* exists
o	Current page number is odd
e	Current page number is even
t	Formatter is *troff*
n	Formatter is *nroff*
v	Never. (**v** is used by other roff variants.)

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. The number of possible environments is only limited by available memory.

Inside each environment, a smaller set of parameters forms the *inline environment*. These are: the current and previous point size, as set by **ps** and **\s**; the current and previous font, as set by **ft** and **\f**; the control and nobreak control character, as set by **cc** and **c2**, respectively; the optional hyphenation character, as set by **hc**; the hyphenation flag, as set by **hy**; the tab and leader repetition characters, as set by **tc** and **lc**, respectively; and the default line breaking penalty, as set by **\J**. The inline environment is pushed by **\@{**, meaning that the current values of these parameters are saved. When a **\@}** occurs, the last pushed inline environment is popped, meaning that the previous values of the parameters are restored. Inline environments can be nested to arbitrary depths. They differ from **\s0** and **\fP** since they form a stack. Thus, the current font is "B" again after the sequence "\fB...\@{\fR...\@{\fI...\@}...\@}", but "I" after the sequence "\fB...\fR...\fI...\fP...\fP".

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
.ev *name*	*name*=0	previous	–	Environment switched to environment *name*. Switching is done in push-down fashion so that restoring a previous environment *must* be done with **.ev** rather than specific reference. Note that what is pushed down and restored is the environment *name,* not its contents. The name of the current environment is available in the **.ev** number register.
.evc *name*		–	–	Copy the environment *name* to the current environment. The temporary state of the current environment is reset, and incompletely filled lines are discarded.

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. 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

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
.so *filename*		–	–	Switch source file. The top input (file reading) level is switched to *filename*. When the new file ends, input is again taken from the original file. It is a fatal error if *filename* cannot be opened. **so**'s may be nested.

.pso *string*	–	–	Execute *string* and read its standard output as text input.
.nx *filename*	end-of-file	–	Next file is *filename*. The current file is considered ended, and the input is immediately switched to *filename*.
.sy *string*	–	–	Execute program from *string*, which is the rest of the input line. The output is not collected automatically. The number register **$$**, which contains the process id of the *troff* process, may be useful in generating unique filenames for output.
.pi *string*	–	–	Pipe output to *string*, which is the rest of the input line. This request must occur *before* any printing occurs; typically it is the first line of input.
.cf *filename*	–	–	Copy contents of file *filename* to output, completely unprocessed. The file is assumed to contain something meaningful to subsequent processes.
.open *stream filename*	ignored	–	Open *filename* for writing while truncating existing contents and associates *stream* with it for latter use with **write** etc.
.opena *stream filename*	ignored	–	Like **open** but appends to *filename* instead of truncating an existing file.
.write *stream text*	ignored	–	Write *text* to file *stream*, which must have been obtained by a previous **open** request. *text* is interpreted in *copy mode.*
.writec *stream text*	ignored	–	Like **write** but does not write a terminating newline.
.writem *stream xx*	ignored	–	Write contents of string, macro, or diversion *xx*. No newline is appended, so if *xx* is a string, the output does not terminate with a newline.
.close *stream*	–	–	Close the file *stream*, which must have been obtained by a previous **open** request.

20. Miscellaneous

The **\V***x*, **\V(***xx*, or **\V[***xxx***]** escape sequence causes the value of the *x*, *xx*, or *xxx* environment variable, respectively, to be printed. It is interpreted in *copy mode.*

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.
.lpfx *string*	off	off	E	Set the line prefix to *string*. ‘‘Whenever a new output line is ‘‘started, *string* is then printed at its beginning. Font and size ‘‘settings as well as automatic ligatures in *string* are retained ‘‘regardless of later changes. This request is mainly useful to ‘‘implement a historic form of block quotation. The current ‘‘value is available in the **.lpfx** register.’’
.tm *string*	–	newline	–	After skipping initial blanks, *string* (rest of the line) is read in *copy mode* and written on the standard error.
.tmc *string*	–	newline	–	Like **tm** but does not write a terminating newline.

.nop *remainder of line*		–	–	Use *remainder of line* as input.
.chomp *remainder of line*		–	–	Use *remainder of line without newline* as input.
.ab *string*	–	newline	–	After skipping initial blanks, *string* (rest of the line) is read in *copy mode* and written on the standard error. *troff* or *nroff* then exit.
.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.
.lf *N filename*		–	–	Set line number to *N* and filename to *filename* for purposes of subsequent error messages, etc. The number register [sic] **.F** contains the name of the current input file, as set by command line arguments, **so**, **nx**, or **lf**. The number register **.c** contains the number of input lines read from the current file, again perhaps as modified by **lf**.
.pm *t*	–	all	–	Print macros. The names and sizes of all of the defined macros and strings are printed on the standard error; if *t* is given, only the total of the sizes is printed. The sizes is given in *blocks* of 128 characters.
.fl	–	–	B	Flush output buffer. Force output, including any pending position information.

21. Output and Error Messages, Debugging.

21.1. Output Messages. The output from **tm**, **pm**, and the prompt from **rd**, as well as various *error* messages are written onto the standard error. The latter is different from the *standard output*, where formatted text goes. By default, both are written onto the user's terminal, but they can be independently redirected. An error message additionally includes the line number where the error occurred, the current input file name, the current physical output page number (if any), and the names of the macros in the frames of the current execution stack. The **errprint** request allows to write custom messages in the same format.

21.2. Warnings. nroff and *troff* provide a mechanism to enable or disable warnings for several categories. When one of the selected conditions occurs, a *warning* message is written in the same format as an error message, but processing continues.

Bit	Name	Description
0	**none**	No warnings at all.
1	**char**	Warn when unknown character names like **\(xx** or **\[xxx]** are found.
2	**number**	Warn when illegal numerical expressions occur.
4	**break**	Warn when a line in **.ad b** mode cannot be adjusted.
8	**delim**	Warn when a delimiter to an escape sequence is missing.
16	**el**	Warn when a **el** request is found without a matching **ie**.
32	**scale**	Warn when an undefined scale indicator is used in a numerical expression.
64	**range**	Warn when an argument is out of range.
128	**syntax**	Warn about questionable syntax in numerical expressions.
256	**di**	Warn when a **di** request is executed but no diversion is currently active.
512	**mac**	Warn when an undefined request, macro or string is called.
1024	**reg**	Warn when an undefined number register is accessed. The number register will be set to zero immediately after the first access so this warning can be printed only once per register name.
4096	**right-brace**	Warn when a **\}** terminates a numerical expression.
8192	**missing**	Warn when a required argument to a request is missing.

Bit	Name	Description
16384	**input**	Warn when illegal byte sequences or characters that have no known PostScript equivalent are read.
32768	**escape**	Warn when an undefined escape sequence is used.
65536	**space**	Warn when an unknown long request name is used, but its first two characters form a known regular request. The regular request is then executed in extension level 2, or ignored in extension level 3.
131072	**font**	Warn when a font cannot be found. This warning is enabled by default.
	all	All warnings except **di**, **mac**, and **reg**. This may be the best choice when using existing macro packages.
	w	All warnings.

21.3. Errors. 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 ☜ in *troff.* Processing continues 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, along with a list of the macro names currently active. 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.

21.4. Debugging. Strings, macros, and number registers can be *watched*. Whenever a change of a watched object occurs, or when the object is renamed, removed, or aliased, a notification message is printed. It is formatted like an error message and includes the name of the current (or last previous) request, the name of the objects affected, and, in case of a change, the new contents of the object.

When an object is removed, watching is disabled. If an object of the same name is created later, it is not watched unless watching is explicitly enabled for it again.

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
.warn ±*bits*|*name*		*w*	–	Control warning messages, which may be given either numerically as bits or symbolically as names. With a **+** sign, the respective bit or name is enabled in addition to the currently enabled categories; with a **–** sign, it is disabled. Omitting the sign sets the categories exactly to the given bit or name. **.warn 0** disables all warnings. The **.warn** register represents the currently activated warning categories as bits.
.spreadwarn *N*		toggle	**m**	Set or toggle a limit that causes a warning to be printed when it is exceeded by the adjustment that is computed for the current output line in **ad b** mode. The limit is initially 3 m, but the warning message is disabled. Calling this request without an argument toggles the warning message; calling it with an argument enables the warning and sets the limit to *N* (default scale m).
.errprint *string*		newline	–	Print *string* like an error message.
.watch *xx*	off	ignore	–	Notify on change of string or macro *xx*. If *xx* does not exist, it is created as an empty macro in order to watch future changes.
.unwatch *xx*	off	ignore	–	Disable notification for string or macro *xx*.

.watchlength *N*		ignore	–	On change, report the contents of macros and strings up to length *N*. When *N* is zero or small, printing of macro and string contents is disabled.
.watchn *R*	off	ignore	–	Notify on change of register *R*. If *R* does not exist, it is created in order to watch future changes. No effect on read-only registers and some of the predefined general registers.
.unwatchn *R*	off	ignore	–	Disable notification for register *R*.

22. Color Support

While *troff* does not support colors directly, *dpost* is able to embed arbitrary PostScript color instructions using the **\X´SetColor:** *color´* escape sequence. Possible values for *color* include:

– ‘‘*named-color*’’, e.g. ‘‘red’’. Named colors (RGB only) must be listed in the ‘‘colordict’’ dictionary in file **/usr/ucblib/doctools/font/devps/postscript/color.ps**. Every *color* argument that begins with a letter is treated as a named color.

– ‘‘*red green blue* **rgb**’’, e.g. ‘‘.2 .3 .4 rgb’’ (**rgb** is an abbreviation for the PostScript **setrgbcolor** operator)

– ‘‘*hue saturation brightness* **hsb**’’, e.g. ‘‘.5 .6 .7 hsb’’ (**hsb** is an abbreviation for the PostScript **sethsbcolor** operator)

– ‘‘*cyan magenta yellow black* **cmyk**’’, e.g. ‘‘.1 .2 .3 .4 cmyk’’ (**cmyk** is an abbreviation for the PostScript **setcmykcolor** operator)

– ‘‘*gray* **setgray**’’, e.g. ‘‘.5 setgray’’

– ‘‘[$*setcolorspace*] *comp1 comp2* ... *compn* **setcolor**’’, where $*setcolorspace* may be a PostScript procedure defined in the setup section using a ‘\X´PSSetup: $setcolorspace {...} bind def´’ escape sequence. This parameter is required if the color space changes in the document. Otherwise, it may be sufficient to install the color space once in the setup section.

Each parameter must be a number in the range between 0.0 and 1.0. In the most general case, the value of the *color* argument is passed to the PostScript output without interpretation.

Both the text and background color can be selected. A *color* argument of ‘‘*color1*’’ **on** ‘‘*color2*’’ prints text in *color1* on a background in *color2*.

The initial color is black in the *DeviceGray* color space, i.e. the same as ‘‘0 setgray’’. Once a color is in effect, it is re-installed at the top of each output page.

The **–mcolor** macro package adds another access method to the capabilities of color selection and reverse video printing. It includes the following macro:

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
.CL *color text*		RGB black	–	Print *text* in *color*. No arguments restores a default color (black in RGB color space; note that this is different from the initial color). If *text* is omitted the selected *color* remains in effect until another *color* is selected. If two arguments are given the *text* is printed in *color* and then the default color is restored.

23. Picture Inclusion

dpost can be advised to include other PostScript documents into the output it generates. In general, these documents should be EPS (Encapsulated PostScript) files; inclusion of more general PostScript documents, especially if they consist of multiple pages, will usually not lead to acceptable results.

If a PostScript file lacks page-delimiting comments, the entire file is included. If no **%%BoundingBox** or **%%HiResBoundingBox** comment is present, the picture is assumed to fill an 8.5×11-inch page. Nothing prevents the picture from being placed off the page.

dpost handles DSC font comments in the included files, but it can only supply glyph data if a path to the respective font file has previously been specified with a *troff* **fp** request. It is not necessary that the font is otherwise used in the *troff* input text. If *dpost* cannot retrieve a matching font by this mechanism, it indicates this in the **%%DocumentNeededResources** comment so that a print manager at a later production stage may include the missing data.

An example how to combine the following requests and macros is:

```
.psbb picture.eps
.nrf scale .25
.nrf y (\n[ury]p–\n[lly]p)*\n[scale]
.nrf x (\n[urx]p–\n[llx]p)*\n[scale]
.PI picture.eps "\nyu,\nxu"
.sp \nyu
picture description
```

troff includes a request to assist the inclusion of EPS files:

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
.psbb *filename*		–	–	Read the **%%HiResBoundingBox** DSC comment, or, if no such comment is found, **%%BoundingBox**, from the PostScript document *filename* and assign the lower left *x* coordinate to the floating-point register **llx**, the lower left *y* coordinate to **lly**, the upper right *x* coordinate to **urx**, and the upper right *y* coordinate to **ury**. All values are in points. If an error occurs, the registers are set to zero.

The **–mpictures** macros insert the necessary advices to *dpost* to include PostScript pictures into *troff* documents. The macros are:

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
.BP *source height width position offset flags label*				Define a frame and place a picture in it. The arguments are: *source* Name of a PostScript picture file, optionally suffixed with (*n*) to select page number *n* from the file (first page by default). *height* Vertical size of the frame, default **3.0i**. This argument is interpreted as a value in inches unless it ends with a **u** scale indicator. *width* Horizontal size of the frame, current line length by default. This argument is interpreted as a value in inches unless it ends with a **u** scale indicator. *position* **l** (default), **c**, or **r** to left-justify, center, or right-justify the frame. *offset* Move the frame horizontally from the original *position* by this amount, default **0i**. *flags* One or more of: **a** *d* Rotate the picture clockwise *d* degrees, default *d* =90. **o** Outline the picture with a box. **s** Freely scale both picture dimensions. **w** White out the area to be occupied by the picture.

l, **r**, **t**, **b** Attach the picture to the left right, top, or bottom of the frame.

label Place *label* at distance **1.5v** below the frame.

If there is room, **BP** fills text around the frame. Everything destined for either side of the frame goes into a diversion to be retrieved when the accumulated text sweeps past the trap set by **BP** or when the diversion is explicitly closed by **EP**.

BP is not recommended for text filling in **ad p** mode since its trap-based mechanism may result in less optimal output, and since it is not possible to define the shape of a complete paragraph with the information it has. Use a combination of **psbb**, **pshape**, and **PI** instead.

.EP – – – End a picture started by **BP**; **EP** is usually called implicitly by a trap at frame bottom. A picture and associated text silently disappear if a diversion trap set by **BP** is not reached. Call **EP** at the end of the document to retrieve it.

.PI *source height***,***width***,***yoffset***,***xoffset flags* This low-level macro, used by **BP**, can help do more complex things. The two arguments not already described are:

xoffset Offset the frame from the left margin by this amount, default **0i**. This argument is interpreted as a value in inches unless it ends with a **u** scale indicator.

yoffset Offset the frame from the current baseline, measuring positive downward, default **0i**. This argument is interpreted as a value in inches unless it ends with a **u** scale indicator.

24. Special Features for PDF Documents

24.1. The basics. *troff* does not directly generate PDF documents; the *dpost* postprocessor generates PostScript output which can be converted to PDF by utilities like Ghostscript's *ps2pdf* or Adobe Distiller. But it is possible to include special advices to this conversion program in PostScript using the **pdfmark** operator. Such advices are generated automatically by some *troff* requests, e.g. by **cropat** (§3). Other advices can be given explicitly using the **\X´PS:**...´ or **\X´PDFMark:**...´ escape sequences.

24.2. Preparations. When generating PDF files, it is especially important to set the paper format using the **mediasize** request (§3). This is because calculations in PDF documents are generally performed in relation regard to the page bottom, while *troff* performs its calculations in relation to the page top. Failing to set the paper format correctly will thus usually result in vertical displacements of PDF-specific elements.

24.3. Specifying document description items. PDF documents can include meta-data about author, title etc. To generate such data, use the **PDFMark** device command of *dpost* with the *troff* **\X** escape sequence, e.g.

\X´PDFMark: Author *My Name´*
\X´PDFMark: Keywords *Typesetting, PDF documents´*
\X´PDFMark: Subject *troff, dpost, and pdfmark´*
\X´PDFMark: Title *Special features for PDF documents´*
.br

Unicode characters are accepted in these text strings. Note that whitespace and newlines surrounding **\X** escape sequences are considered to be input text by *troff,* and need a *break* before they are output. To avoid inserting superfluous spaces or line breaks, specify this information before an initial **.sp |\n[topmargin]u** request or the like in the document.

24.4. Direct use of the pdfmark operator. In cases where *troff* or *dpost* do not include an explicit mechanism for PDF features, it is possible to call the **pdfmark** operator directly. Examples are:

\X´PS: [{Catalog} << /ViewerPreferences << /DisplayDocTitle true >> >> /PUT pdfmark\'
This causes the PDF viewer to print the document title (as in §24.3) in the application title bar instead of the PDF file name.

\X´PS: [/PageMode /UseOutlines /DOCVIEW pdfmark´
This causes the PDF viewer to display the bookmarks toolbar when the document is opened. Other interesting values are **/UseNone** (the default), **/UseThumbs** (display page thumbnails), and **/FullScreen** (open the document in full-screen mode).

\X´PS: [/PageMode /Page *N* **/View [/XYZ null null null] /DOCVIEW pdfmark´**
Page *N* is displayed instead of the first page when the document is opened.

\X´PS: [{Catalog} << /PageLayout /TwoPageRight >> /PUT pdfmark´
With this command, the PDF viewer displays two pages at once. Other interesting values are **/SinglePage** (display one page at once), **/OneColumn** (display one page in continuous mode), and **/TwoColumnRight** (display two pages in continuous mode).

\X´PS: [/Label (*text***) /PAGELABEL pdfmark´**
The given ASCII *text* is shown next to the page number of the current page in the PDF viewer toolbar. This is particularly useful to implement roman page numbers in PDF documents in combination with the **af** request.

24.5. Creating bookmarks. Bookmarks are usually shown by the PDF viewer at the left of the window. When you create PDF files longer than a few pages for viewing on the screen, you should include bookmarks for each chapter or section because they enable the reader to navigate much more conveniently.

Start with a bookmark for the title page. "0" is the level of the bookmark in the tree structure, and the following arguments form the name of the bookmark: **\X´PDFMark: Bookmark** *0 Title´*. Similar to the description specifications above, this bookmark should appear before the top margin of the title page.

When generating bookmarks for chapters and sections, it is usually most practical to include the command in the chapter/section macros:

```
.de CH
.	bp
\v´|–1v–4p´\\X´PDFMark: Bookmark 0 \\$1´
.	sp |\\n[topmargin]u
.	ce
.	sp
..
.de SH
.	ce
\\$1\v´–1v´\\X´PDFMark: Bookmark 1 \\$1´
.	sp
..
```

Unless the bookmark command occurs at the top of the page, it refers to the baseline of the text; this is why a **\v´–1v´** command occurs before it in the section macro. **\v´–1v–4p´** at the top of the page is a special value that causes the upper left corner to be shown.

The level of the first bookmark in a document must be of level 0. The levels of following bookmarks must be either

– one higher than the level of the previous bookmark. The new bookmark then becomes a child of the previous bookmark.

– equal to the level of the previous bookmark. In this case, both bookmarks are grouped below the first previous bookmark of the first higher level, or at the top level for level 0.

– lower than the level of the previous bookmark. This terminates the list of children of the higher levels.

As an example, the following level structure is legal, and causes the PDF viewer to group the bookmarks as shown:

0	title page
0	Chapter 1
• ——1	Section 1.1
• ——1	Section 1.2
| •———2	Subsection 1.2.1
| •———2	Subsection 1.2.2
| •———2	Subsection 1.2.3
• ——1	Section 1.3
•———2	Subsection 1.3.1
•———2	Subsection 1.3.2
0	Chapter 2
• ——1	Section 2.1
0	Index

In case of an illegal structure in which the level of a bookmark is raised by more than one above its predecessor, *dpost* emits a warning and assigns the highest legal level. However, a garbled document structure may result:

0	
• ——2	☜ the first level 2 is corrected to 1 by *dpost*
•———2	so the second bookmark becomes a child of the first

Thus such *dpost* warnings should usually be taken seriously, and the document should be corrected.

An alternate form of bookmarks, **\X´PDFMark: BookmarkClosed** ...´, is available. The syntax is the same, but the initial view in the tree structure is collapsed, i.e. no children are shown by default. It the bookmark has no children, there is no difference to a regular bookmark.

24.6. Links. PDF documents can contain links that cause the viewer to jump to a certain location when the user clicks on an area of the page, as well as links to external documents in URI form. In *troff,* such links can be built as follows:

The **\A**´*string*´ escape sequence defines an anchor, i.e. a location to jump to, with the name *string* (consisting of ASCII characters).

The actual link is built using two **\T** escape sequences surrounding the text that forms the area to click on, e.g.: **\T**´*string*´*text of link***\T**. *string* must correspond to an anchor anywhere in the document.

An URI link is built likewise using two **\W** escape sequences: **\W**´*uri*´*text of link***\W**. The *uri* part is not interpreted by *troff*, but just written to the generated output. Typically, this will be a link to a web page, as in <\W’http://n-t-roff.github.io/heirloom/doctools.html’http://n-t-roff.github.io/heirloom/doctools.html\W>.

The appearance of links can be changed; links are normally surrounded by an 1 point wide blue border. The color can be set using **\X’SetLinkColor:** *red green blue***’**, where *red, green, and blue* are values between 0 and 1. The border can be set using **\X’SetLinkBorder:** *bx by width***’**, where *bx* and *by* define the horizontal and vertical corner radius, respectively, and *width* defines the width.

The border style can be changed with the **\X'SetBorderStyle:** *arguments***'** command. The *arguments* are the same as for the **/BS** pdfmark operator or the LaTeX **hyperref pdfborderstyle={}** variable.

Likewise, **SetULinkColor**, **SetULinkBorder**, and **SetUBorderStyle** are available for URI links.

E.g. for this document the commands

```
\X'SetBorderStyle: /S/U/W 0.1'
\X'SetUBorderStyle: /S/U/W 0.1'
```

are used.

25. *groff* Compatibility

Heirloom *troff* provides most of the extensions to the *troff* language introduced in *groff* [10]. Consequently, it is possible to create documents, macro sets, and preprocessors that can be used with both Heirloom *troff* and *groff* and use functionality beyond the features that were supported by traditional *troff.*

Important differences to *groff* are: The concept of an *input level* regarding delimiters in escape sequences and macro arguments is not supported. Control characters are recognized at the beginning of a line even if preceded by escape sequences that do not result in formatting stream objects, such as **\f**, **\s**, or **\;**. Font handling, color support, picture inclusion, and PDF structuring are realized using different mechanisms. The *dpost* post-processor does not recognize the **\X´ps:**...´ escape sequence (or **x X ps:** command, respectively) that is used for pass-through PostScript with the *grops* post-processor of *groff* (*dpost* accepts **\X’PS:**...´ and **x X PS:**); the PostScript output generated by *dpost* is very different to that generated by *grops. dpost* accepts the *groff* drawing command extensions and sets the horizontal and vertical positions accordingly, but otherwise ignores line width, color, and fill specifications.

25.1. Conditional groff compatibility. A request is available to control additional functions for *groff* compatibility:

Request Form	***Initial Value***	***If No Argument***	***Notes***	***Explanation***
.cp *N*	off	–	–	Enable *groff* compatibility mode. This is the name of *groff’s* own compatibility request with adapted semantics: Regardless of the argument, *groff* compatibility mode is activated. If *N*=0, compatibility with traditional *troff* is decreased, and Heirloom *troff* extension level 3 is set. If *N*≠0 or missing, compatibility with traditional *troff* is improved, and Heirloom *troff* extension level 1 is set. Thus **.cp 0** results in maximum *groff* compatibility.

The **cp** request sets the general number register **.g** to 1 in *groff* compatibility mode and to 0 otherwise. The general number register **.C** is only assigned by the **.cp** request and corresponds to its argument.

Any use of the **xflag** request disables *groff* compatibility and accordingly sets the **.g** register to 0.

The **.X** read-only number register holds the current extension level after **cp** as usual. It can thus be used to determine whether running under *groff* or in the *groff* compatibility mode of Heirloom *troff.* Since they are read-write, any convenient value can then be assigned to the **.C** and **.g** registers.

Nevertheless it is not recommend to do so without reason. For instance macro packages read \n(.g to test if they are processed with groff. Setting **.g** to **1** with the **.nr** request also has some side effects in Heirloom *troff*:

- *groff*’s notation for accessing symbols with **\[char***n***]** and **\[u***XXXX***]** are enabled.
- The escape **\%** marks optional hyphenation points inside a word or suppresses hyphenation prefixed to a word (*even when this word containes dashes*). A problem in macro packages is that it cannot be guaranteed that no character is before word (e.g. quotes or parentheses). *groff* allows to force that **\%** acts as a hyphenation suppression escape when it follows the zero-width characters **\)** or **\&**. This does not work with traditional *troff*, unfortunately. The command **.nr .g 1** enables the *groff* behaviour.
- The algorithm *groff* uses for the three part title request **.tl** leads to a different placement of the middle title element with *nroff* under certain conditions. This is emulated by setting **.g** to 1.
- *groff*’s left italic correction escape **\,** is removed from the input instead of producing a ‘**,**’.

The **.x** and **.y** registers are read-write in *groff* compatibility mode; they correspond to the emulated *groff* version number. The **cp** request sets them to 1 and 18, respectively.

The **.k** number register behaves differently in *groff* compatibility mode: If the preceding text character was a newline, the width of a space character is added to the value. If the previous line was interrupted with **\c**, the length of the partially collected word (as in the **.kc** number register) is part of the value.

The space width always defaults to the value obtained from the font metrics file in *groff* compatibility mode; the **spacewidth** request is not effective.

If the file specified with a **so** request cannot be opened, processing continues with the current file in *groff* compatibility mode.

A control or escape character written in a diversion has no special meaning if the diversion is re-read in *groff* compatibility mode.

Unless a string is interpolated with explicit arguments, the arguments to the surrounding macro instance remain visible and can be referenced by the **\$** escape sequence inside the string in *groff* compatibility mode.

Any call to the **cp** request activates the following *groff* compatibility escape sequences; any call to the **xflag** request disables them.

The **\A***'string'* escape sequence checks whether *string* is acceptable as the name of a string, macro, number register, or font, and evaluates to "1' if it does and to "0" otherwise. The Heirloom *troff* anchoring escape sequence **\A** is not available in *groff* compatibility mode.

The \/ escape sequence inserts an italic correction, i.e. a small piece of horizontal motion (1/12 em) that should be sufficient to separate an italic character from a following roman character. Similarly, **\,** adds a left italic correction, always a zero motion, that should be sufficient to separate a roman character from an immediately following italic character. Both exist to provide basic *groff* compatibility only. It is otherwise recommended that the **kernpair** request is used for these purposes; it allows a more exact optical separation since the shape of both characters can be taken into account and also does not need to be be positioned directly in the input text at every occasion.

The **\D´p** ...´ polygon drawing escape sequence is altered such that the path is always closed, i.e. if the last line part does not return to the starting position, an additional line is added that does.

The escape sequences **\F***x*, **\F(***xx*, **\F[***xxx***]**, **\m***x*, **\m(***xx*, **\m[***xxx***]**, **\M***x*, **\M(***xx*, **\M[***xxx***]**, are read but discarded in *groff* compatibility mode since the corresponding concepts of font families and built-in color support are foreign to Heirloom *troff.* All of them generate a warning of the **escape** category.

25.2. groff compatibility macros. As an additional aid in formatting documents that had originally been written with *groff* in mind, Heirloom *troff* provides the **–mg** compatibility macro package. Specifically:

.cp 0 is executed, so *troff* is operated in *groff* compatibility mode at extension level 3.

The locale is set to **en_US**; this assumes that input to *groff* is conventionally in the ISO-8859-1 encoding.

The hyphenation language is set to **en_US**. Since both *groff* and Heirloom *troff* then use the same hyphenation algorithm and patterns derived from T_EX, it can be assumed that words are subject to hyphenation at the same points.

The **de1**, **am1**, **ds1**, and **as1** *groff* requests are emulated using **de**, **am**, **ds**, and **as**, respectively. It is not expected that the *groff* compatibility macros are used to format documents that require compatibility with traditional *troff.*

The **ftr** request is removed since the *groff* request with the same name provides completely different semantics.

fallback is used to emulate **fspecial**; these requests should be compatible. No emulation is provided for the **special** request; all other fonts are searched for missing characters in Heirloom *troff* by default.

track is used to emulate **tkf**. These requests are not completely compatible: **track** does not affect the last character on a line and thus does not alter the visual length of lines like **tkf** does; **track** is applied when formatting an object defined with the **char** request; **track** needs to remain in effect until all affected characters have been physically output. Nevertheless, the replacement should suffice for most applications.

groff characters with two-character names are mapped to PostScript character names using the **char** request.

The following macro is also provided as an emulation for the corresponding *groff* request:

Request Form	*Initial Value*	*If No Argument*	*Notes*	*Explanation*
.mso *name*	–	ignored	–	Include the macro package *name*. If the environment variable **GROFF_TMAC_PATH** is set, each of the colon-separated directories in it is checked for the files *name*, *name*.tmac, m*name*.tmac, and tmac.*name*. After this, the standard *troff* macro directories are checked in the same way. Once a file has been found, it is read in as with the **so** request, and the search stops.

26. Output Language.

troff produces its output in a language that is independent of any specific output device, except that the numbers in it have been computed on the basis of the resolution of the device, and the sizes, fonts, and characters that that device can print. Nevertheless it is quite possible to interpret that output on a different device, within the latter's capabilities.

s*n*	set point size to *n*, fractional parts (if any) ignored
s–23*d.d*	set point size to *d.d*
f*n*	set font to *n*
c*c*	print character *c*
C*name*	print the character called *name*; terminate *name* by white space
CPS*name*	print the character with the given PostScript *name*
N*n*	print character *n* on current font
H*n*	go to absolute horizontal position *n* ($n \geq 0$)
V*n*	go to absolute vertical position *n* ($n \geq 0$, down is positive)
h*n*	go *n* units horizontally; $n<0$ is to the left
v*n*	go *n* units vertically; $n<0$ is up
nnc	move right *nn*, then print ASCII character *c*; *nn* must be exactly 2 digits
p*n*	new page *n* begins—set vertical position to 0
n*b a*	end of line (information only—no action); *b* = space before line, *a* = after
w	paddable word space (information only—no action)
D*c* ...\n	graphics function *c*; see below
x ...\n	device control functions; see below
# ...\n	comment

All position values are in units. Sequences that end in digits must be followed by a non-digit. Blanks, tabs and newlines may occur as separators in the input, and are mandatory to separate constructions that would otherwise be confused. Graphics functions, device control functions, and comments extend to the end of the line they occur on.

The device control and graphics commands are intended as open-ended families, to be expanded as needed. The graphics functions coincide directly with the **\D** sequences:

Dl *dh dv*	draw line from current position by *dh, dv*
Dp *a b c d ...*	draw polygon, i.e. a line to *a, b*, then to *c, d*, then ...
Dc *d*	draw circle of diameter *d* with left side at current position
De *u v*	draw ellipse of diameters *u* and *v*
Da *a b c d*	draw arc from current position to *a*+*c*, *b*+*d*, with center at *a*, *b* from current position
D~ *a b c d...*	draw B-spline from current position by *a*, *b*, then by *c*, *d*, then by *c*, *d*, then ...
Dz *a b c d...*	for any other *z* is uninterpreted

In all of these, *x, y* is an increment on the current horizontal and vertical position, with down and right positive. All distances and dimensions are in units.

The device control functions begin with **x**, then a command, then other parameters.

x T *s*	name of typesetter is *s*
x r *n h v*	resolution is *n* units/inch; *h* = minimum horizontal motion, *v* = minimum vertical motion
x i	initialize
x f *n s*	mount font *s* on font position *n*
x f *n s filename flags*	*filename* contains metrics for font *s* on position *n* using *flags*
x p	pause—can restart
x s	stop—done forever
x t	generate trailer information, if any
x H *n*	set character height to *n*, fractional parts (if any) ignored
x H −23 *d.d*	set character height to *d.d*
x S *n*	set slant to *n*
x X *any*	generated by the **\X** and **\Y** functions
x X Anchor *y,x id*	specify PDF link anchor (generated by the **\A** request)
x X Anchor *id*	specify HTML link anchor
x X BleedAt *L T W H*	generated by the **bleedat** request
x X CropAt *L T W H*	generated by the **cropat** request
x X HorScale *percent*	scale letters horizontally by *percent* (with the **letadj** request)
x X LC_CTYPE *name*	sets the LC_CTYPE locale to *name*
x X Link *x1,y1,x2,y2 id*	specify PDF link (generated by the **\T'***id***'** request)
x X Link *id*	begin HTML link (generates ****)
x X Link	end HTML link (generates ****)
x X PaperSize *W H n*	generated by the **mediasize** and **papersize** requests *n* is non-zero for **mediasize**
x X PS *command*	embed PostScript *command*
x X PSSetup *command*	embed PostScript *command* in global setup section
x X SetColor *color*	change printing *color*
x X SupplyFont *font file*	supply data from *file* for PostScript *font*
x X Sync	instructs to flush position and font information immediately
x X Track *n*	gives a hint that following characters are tracked by *n* units interpretation is optional; following positions are not changed
x X TrimAt *L T W H*	generated by the **trimat** request
x X ULink *x1,y1,x2,y2 URL*	specify PDF URL link (generated by the **\W'***id***'** request)
x X ULink *URL*	begin HTML URL link (generates ****)
x X ULink	end HTML URL link (generates ****)
x *any*	to be ignored if not recognized

Subcommands like "i" may be spelled out like "init".

The commands **x T**, **x r** ..., and **x i** fonts must be mounted before they can be used; **x s** comes last. There are no other order requirements.

The following is the output from "hello, world" for a typical printer, as described in §27:

```
x T ps
x res 72000 1 1
x init
V0
p1
x font 1 R /usr/ucblib/doctools/font/devps/R.afm 4
x font 2 I /usr/ucblib/doctools/font/devps/I.afm 4
x font 3 B /usr/ucblib/doctools/font/devps/B.afm 4
x font 4 BI /usr/ucblib/doctools/font/devps/BI.afm 4
x font 5 CW /usr/ucblib/doctools/font/devps/CW.afm 4
x font 6 H /usr/ucblib/doctools/font/devps/H.afm 4
```

```
x font 7 HB /usr/ucblib/doctools/font/devps/HB.afm 4
x font 8 HX /usr/ucblib/doctools/font/devps/HX.afm 4
x font 9 S1 /usr/ucblib/doctools/font/devps/S1.afm 516
x font 10 S /usr/ucblib/doctools/font/devps/S.afm 1028
s10
f1
x X LC_CTYPE de_DE.utf8
H72000
V12000
ch
h5000ce
h4440cl
h2780cl
h2780co
h5000c,
wh5830cw
h7120co
h5000cr
h3330cl
h2780cd
n12000 0
x trailer
V792000
x stop
```

troff output is normally not redundant; size and font changes and position information are not included unless needed. Nevertheless, each page is normally self-contained, for the benefit of postprocessors that re-order pages or process only a subset. The **x X PSSetup** command is an exception from this rule; it is included only once at the point where the corresponding **\X**'**PSSetup:** ...' escape sequence occurs.

27. Device and Font Description Files

The parameters that describe a output device *name* are read from the directory **/usr/ucblib/doctools/font/dev***name*, each time *troff* is invoked. The device name is provided by default, by the environment variable **TYPESETTER**, or by a command-line argument **–T***name*. The default device name is **ps**, for PostScript output at a resolution of 72 000 dpi. The pre-defined string **.T** contains the name of the device. The **–F** command-line option may be used to change the default directory.

27.1. Device description file. General parameters of the device are stored, one per line, in the file **/usr/ucblib/doctools/font/dev***name***/DESC**, as a sequence of names and values. *troff* recognizes these parameters, and ignores any others that may be present for specific drivers:

fonts *n F G H ... Z*
sizes *s t ...* **0**
res *n*
hor *n*
vert *n*
unitwidth *n*
charset
list of multi-character character names (optional)

The F, G, ... are font names to be initially mounted. The list of sizes is a set of integers representing some or all of the legal sizes the device can produce, terminated by a zero. The **res** parameter gives the resolution of the machine in units per inch; **hor** and **ver** give the minimum number of units that can be moved horizontally and vertically.

Character widths for each font are assumed to be given in machine units at point size **unitwidth**. (In other words, a character with a width of *n* is *n* units wide at size **unitwidth**.) All widths are integers at all sizes.

A list of valid character names may be introduced by **charset**; the list of names is optional.

A line whose first non-blank character is **#** is a comment. Except that **charset** must occur last, parameters may appear in any order.

Here is a subset of the **DESC** file for a typical Postscript printer:

```
# Description file for Postscript printers.

fonts 10 R I B BI CW H HB HX S1 S
sizes 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
   24 25 26 27 28 29 30 31 32 33 34 35 36 38 40 44 48 54 60 72 0
anysize
allpunct
afmfonts
lc_ctype
res 72000
hor 1
vert 1
unitwidth 1
charset
hy ct fi fl ff Fi Fl dg em 14 34 12 en aa
ga ru sc dd -> br Sl ps cs cy as os =. ld
rd le ge pp -+ ob vr rs dq
sq bx ci fa te ** pl mi eq |= *A *B *X *D
*E *F *G *Y *I *K *L *M *N *O *P *R *H *S *T *U *W
*C *Q *Z ul rn *a *b *x *d *e *f *g *y *i *k
*l *m *n *o *p *h *r *s *t *u *w *c *q *z
```

27.2. Font description files. Note: This description applies to the old *troff* device-independent font format. The current version of *troff* usually reads font metrics directly from Type 1, OpenType, or TrueType font files, as described for the **fp** request (§2) and in the separate fonts manual available from the project home page.

Each font is described by an analogous description file, which begins with parameters of the font, one per line, followed by a list of characters and widths. The file for font *F* is **/usr/ucblib/doctools/font/dev***name***/***F*.

name *str*	name of font is *str*
ligatures ... 0	list of ligatures
spacewidth *n*	width of a space on this font
special	this is a special font
charset	

list of character name, width, ascender/descender, code, tab separated

The **name** and **charset** fields are mandatory; **charset** must be last. Comments are permitted, as are other unrecognized parameters.

Each line following charset describes one character: its name, its width in units as described above, ascender/descender information, and a decimal, octal or hexadecimal value by which the output device knows it (the **\N** "number" of the character). The character name is arbitrary, except that **---** signifies an unnamed character. If the width field contains **"**, the name is a synonym for the previous character. The ascender/descender field is 1 if the character has a descender (hangs below the baseline, like **y**), is 2 if it has an ascender (is tall, like **Y**), is 3 if both, and is 0 if neither. The value is returned in the **ct** register, as computed by the **\w** function (§11.2).

Here are excerpts from a typical font description file for the same Postscript printer.

```
hy      33      0       45          hyphen \(hy
-       "                           - is a synonym for \(hy

Q       72      3       81
```

```
a       44      0       97
b       50      2       98
c       44      0       99
d       50      2       100
y       50      1       121

em      100     0       208
---     44      2       220             English pound currency symbol \N'220'
---     36      0       221             centered dot \N'221'
```

This says, for example, that the width of the letter **a** is 44 units at point size 10, the value of **unitwidth**. Point sizes are scaled linearly and rounded, so the width of **a** will be 44 at size 10, 40 at size 9, 35 at size 8, and so on.

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.

*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).

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

```
.de fo
.nr 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 baseline 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. Note that this table only includes the historical CAT *troff* character set; PostScript devices can usually print a much larger, although font-dependent set of characters.

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 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.

In traditional *troff,* the ASCII characters @, #, ", ´, `, <, >, \, {, }, ˜, ˆ, and _ existed *only* on the special font and were printed as a 1-em space if that font was not mounted. The following characters exist only on the special font. 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
\	\(rs	backslash	τ	\(*t	tau
α	\(*a	alpha	υ	\(*u	upsilon
β	\(*b	beta	φ	\(*f	phi
γ	\(*g	gamma	χ	\(*x	chi
δ	\(*d	delta	ψ	\(*q	psi
ε	\(*e	epsilon	ω	\(*w	omega
ζ	\(*z	zeta	Α	\(*A	Alpha
η	\(*y	eta	Β	\(*B	Beta
θ	\(*h	theta	Γ	\(*G	Gamma
ι	\(*i	iota	Δ	\(*D	Delta
κ	\(*k	kappa	Ε	\(*E	Epsilon

Char	*Input Name*	*Character Name*
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

Char	*Input Name*	*Character Name*
☜	\(lh	left hand
│	\(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)

Available characters from groff_char(7)

The following table list the characters from the **groff_char**(7) manual page which can be used with Heirloom *troff*. Special characters marked with a * are only defined when *troff* is started with option **–mgchar**. If a glyph is actually available depends on the font and output device.

Output	*Input*	*PostScript*	*Unicode*	*Notes*
¡	¡	`\[exclamdown]`	`\U'00A1'`	inverted exclamation mark
¢	¢	`\[cent]`	`\U'00A2'`	currency unit
£	£	`\[sterling]`	`\U'00A3'`	pound sterling
¤	¤	`\[currency]`	`\U'00A4'`	generic currency symbol
¥	¥	`\[yen]`	`\U'00A5'`	Japanese currency symbol
¦	¦	`\[brokenbar]`	`\U'00A6'`	broken bar
§	§	`\[section]`	`\U'00A7'`	section sign
¨	¨	`\[dieresis]`	`\U'00A8'`	dieresis
©	©	`\[copyright]`	`\U'00A9'`	copyright symbol
ª	ª	`\[ordfeminine]`	`\U'00AA'`	feminine ordinal
«	«	`\[guillemotleft]`	`\U'00AB'`	left guillemet [sic]
¬	¬	`\[logicalnot]`	`\U'00AC'`	logical not
®	®	`\[registered]`	`\U'00AE'`	registered mark symbol
¯	¯	`\[macron]`	`\U'00AF'`	overbar accent
°	°	`\[degree]`	`\U'00B0'`	degree
±	±	`\[plusminus]`	`\U'00B1'`	plus-minus sign
²	²	`\[twosuperior]`	`\U'00B2'`	superscript 2
³	³	`\[threesuperior]`	`\U'00B3'`	superscript 3
´	´	`\[acute]`	`\U'00B4'`	acute accent
µ	µ	`\[mu]`	`\U'00B5'`	micro sign
¶	¶	`\[paragraph]`	`\U'00B6'`	end of paragraph marker
·	·	`\[periodcentered]`	`\U'00B7'`	centered period
¸	¸	`\[cedilla]`	`\U'00B8'`	cedilly accent
¹	¹	`\[onesuperior]`	`\U'00B9'`	superscript 1
º	º	`\[ordmasculine]`	`\U'00BA'`	masculine ordinal
»	»	`\[guillemotright]`	`\U'00BB'`	right guillemet [sic]
¼	¼	`\[onequarter]`	`\U'00BC'`	1/4 symbol
½	½	`\[onehalf]`	`\U'00BD'`	1/2 symbol
¾	¾	`\[threequarters]`	`\U'00BE'`	3/4 symbol
¿	¿	`\[questiondown]`	`\U'00BF'`	inverted question mark
À	À	`\[Agrave]`	`\U'00C0'`	A grave
Á	Á	`\[Aacute]`	`\U'00C1'`	A acute
Â	Â	`\[Acircumflex]`	`\U'00C2'`	A circumflex
Ã	Ã	`\[Atilde]`	`\U'00C3'`	A tilde
Ä	Ä	`\[Adieresis]`	`\U'00C4'`	A dieresis
Å	Å	`\[Aring]`	`\U'00C5'`	A ring
Æ	Æ	`\[AE]`	`\U'00C6'`	A+E combined
Ç	Ç	`\[Ccedilla]`	`\U'00C7'`	C cedilla
È	È	`\[Egrave]`	`\U'00C8'`	E grave
É	É	`\[Eacute]`	`\U'00C9'`	E acute
Ê	Ê	`\[Ecircumflex]`	`\U'00CA'`	E circumflex
Ë	Ë	`\[Edieresis]`	`\U'00CB'`	E dieresis
Ì	Ì	`\[Igrave]`	`\U'00CC'`	I grave
Í	Í	`\[Iacute]`	`\U'00CD'`	I acute
Î	Î	`\[Icircumflex]`	`\U'00CE'`	I circumflex
Ï	Ï	`\[Idieresis]`	`\U'00CF'`	I dieresis

Output	*Input*	*PostScript*	*Unicode*	*Notes*
Ð	Ð	\[Eth]	\U'00D0'	E th
Ñ	Ñ	\[Ntilde]	\U'00D1'	N tilde
Ò	Ò	\[Ograve]	\U'00D2'	O grave
Ó	Ó	\[Oacute]	\U'00D3'	O acute
Ô	Ô	\[Ocircumflex]	\U'00D4'	O circumflex
Õ	Õ	\[Otilde]	\U'00D5'	O tilde
Ö	Ö	\[Odieresis]	\U'00D6'	O dieresis
×	×	\[multiply]	\U'00D7'	multiply
Ø	Ø	\[Oslash]	\U'00D8'	O slash
Ù	Ù	\[Ugrave]	\U'00D9'	U grave
Ú	Ú	\[Uacute]	\U'00DA'	U acute
Û	Û	\[Ucircumflex]	\U'00DB'	U circumflex
Ü	Ü	\[Udieresis]	\U'00DC'	U dieresis
Ý	Ý	\[Yacute]	\U'00DD'	Y acute
Þ	Þ	\[Thorn]	\U'00DE'	Thorn
ß	ß	\[germandbls]	\U'00DF'	German double s
à	à	\[agrave]	\U'00E0'	a grave
á	á	\[aacute]	\U'00E1'	a acute
â	â	\[acircumflex]	\U'00E2'	a circumflex
ã	ã	\[atilde]	\U'00E3'	a tilde
ä	ä	\[adieresis]	\U'00E4'	a dieresis
å	å	\[aring]	\U'00E5'	a ring
æ	æ	\[ae]	\U'00E6'	a+e combined
ç	ç	\[ccedilla]	\U'00E7'	c cedilla
è	è	\[egrave]	\U'00E8'	e grave
é	é	\[eacute]	\U'00E9'	e acute
ê	ê	\[ecircumflex]	\U'00EA'	e circumflex
ë	ë	\[edieresis]	\U'00EB'	e dieresis
ì	ì	\[igrave]	\U'00EC'	i grave
í	í	\[iacute]	\U'00ED'	i acute
î	î	\[icircumflex]	\U'00EE'	i circumflex
ï	ï	\[idieresis]	\U'00EF'	i dieresis
ð	ð	\[eth]	\U'00F0'	e th
ñ	ñ	\[ntilde]	\U'00F1'	n tilde
ò	ò	\[ograve]	\U'00F2'	o grave
ó	ó	\[oacute]	\U'00F3'	o acute
ô	ô	\[ocircumflex]	\U'00F4'	o circumflex
õ	õ	\[otilde]	\U'00F5'	o tilde
ö	ö	\[odieresis]	\U'00F6'	o dieresis
÷	÷	\[divide]	\U'00F7'	divide
ø	ø	\[oslash]	\U'00F8'	o slash
ù	ù	\[ugrave]	\U'00F9'	u grave
ú	ú	\[uacute]	\U'00FA'	u acute
û	û	\[ucircumflex]	\U'00FB'	u circumflex
ü	ü	\[udieresis]	\U'00FC'	u dieresis
ý	ý	\[yacute]	\U'00FD'	y acute
þ	þ	\[thorn]	\U'00FE'	thorn
ÿ	ÿ	\[ydieresis]	\U'00FF'	y dieresis
Ð	\(-D*	\[Eth]	\U'00D0'	uppercase eth
ð	\(Sd*	\[eth]	\U'00F0'	lowercase eth
Þ	\(TP*	\[Thorn]	\U'00DE'	uppercase thorn
þ	\(Tp*	\[thorn]	\U'00FE'	lowercase thorn

Output	*Input*	*PostScript*	*Unicode*	*Notes*
ß	\(ss*	\[germandbls]	\U'00DF'	German double s
Ł	\(/L*	\[Lslash]	\U'0141'	L slash
ł	\(/l*	\[lslash]	\U'0142'	l slash
Ø	\(/O*	\[Oslash]	\U'00D8'	O slash
ø	\(/o*	\[oslash]	\U'00F8'	o slash
Æ	\(AE	\[AE]	\U'00C6'	A+E combined
æ	\(ae	\[ae]	\U'00E6'	a+e combined
Œ	\(OE	\[OE]	\U'0152'	O+E combined
œ	\(oe	\[oe]	\U'0153'	o+e combined
	\(IJ	\[IJ]	\U'0132'	I+J combined
	\(ij	\[ij]	\U'0133'	i+j combined
ı	\(.i*	\[dotlessi]	\U'0131'	i without a dot
	\(.j*	\[dotlessj]	\U'0237'	j without a dot
Á	\('A*	\[Aacute]	\U'00C1'	A acute
	\('C*	\[Cacute]	\U'0106'	C acute
É	\('E*	\[Eacute]	\U'00C9'	E acute
Í	\('I*	\[Iacute]	\U'00CD'	C acute
Ó	\('O*	\[Oacute]	\U'00D3'	O acute
Ú	\('U*	\[Uacute]	\U'00DA'	U acute
Ý	\('Y*	\[Yacute]	\U'00DD'	Y acute
á	\('a*	\[aacute]	\U'00E1'	a acute
	\('c*	\[cacute]	\U'0107'	c acute
é	\('e*	\[eacute]	\U'00E9'	e acute
í	\('i*	\[iacute]	\U'00ED'	i acute
ó	\('o*	\[oacute]	\U'00F3'	o acute
ú	\('u*	\[uacute]	\U'00FA'	u acute
ý	\('y*	\[yacute]	\U'00FD'	y acute
Ä	\(:A*	\[Adieresis]	\U'00C4'	A dieresis
Ë	\(:E*	\[Edieresis]	\U'00CB'	E dieresis
Ï	\(:I*	\[Idieresis]	\U'00CF'	I dieresis
Ö	\(:O*	\[Odieresis]	\U'00D6'	O dieresis
Ü	\(:U*	\[Udieresis]	\U'00FC'	U dieresis
Ÿ	\(:Y*	\[Ydieresis]	\U'0178'	Y dieresis
ä	\(:a*	\[adieresis]	\U'00E4'	a dieresis
ë	\(:e*	\[edieresis]	\U'00EB'	e dieresis
ï	\(:i*	\[idieresis]	\U'00EF'	i dieresis
ö	\(:o*	\[odieresis]	\U'00F6'	o dieresis
ü	\(:u*	\[udieresis]	\U'00FC'	u dieresis
ÿ	\(:y*	\[ydieresis]	\U'00FF'	y dieresis
Â	\(^A*	\[Acircumflex]	\U'00C2'	A circumflex
Ê	\(^E*	\[Ecircumflex]	\U'00CA'	E circumflex
Î	\(^I*	\[Icircumflex]	\U'00CE'	I circumflex
Ô	\(^O*	\[Ocircumflex]	\U'00D4'	O circumflex
Û	\(^U*	\[Ucircumflex]	\U'00DB'	U circumflex
â	\(^a*	\[acircumflex]	\U'00E2'	a circumflex
ê	\(^e*	\[ecircumflex]	\U'00EA'	e circumflex
î	\(^i*	\[icircumflex]	\U'00EE'	i circumflex
ô	\(^o*	\[ocircumflex]	\U'00F4'	o circumflex
û	\(^u*	\[ucircumflex]	\U'00FB'	u circumflex
À	\(`A*	\[Agrave]	\U'00C0'	A grave
È	\(`E*	\[Egrave]	\U'00C8'	E grave

Output	*Input*	*PostScript*	*Unicode*	*Notes*
Ì	\(`I*	\[Igrave]	\U'00CC'	I grave
Ò	\(`O*	\[Ograve]	\U'00D2'	O grave
Ù	\(`U*	\[Ugrave]	\U'00D9'	U grave
à	\(`a*	\[agrave]	\U'00E0'	a grave
è	\(`e*	\[egrave]	\U'00E8'	e grave
ì	\(`i*	\[igrave]	\U'00EC'	i grave
ò	\(`o*	\[ograve]	\U'00F2'	o grave
ù	\(`u*	\[ugrave]	\U'00F9'	u grave
Ã	\(~A*	\[Atilde]	\U'00C3'	A tilde
Ñ	\(~N*	\[Ntilde]	\U'00D1'	N tilde
Õ	\(~O*	\[Otilde]	\U'00D5'	O tilde
ã	\(~a*	\[atilde]	\U'00E3'	a tilde
ñ	\(~n*	\[ntilde]	\U'00F1'	n tilde
õ	\(~o*	\[otilde]	\U'00F5'	o tilde
Š	\(vS*	\[Scaron]	\U'0160'	S caron
š	\(vs*	\[scaron]	\U'0161'	s caron
Ž	\(vZ*	\[Zcaron]	\U'017D'	Z caron
ž	\(vz*	\[zcaron]	\U'017E'	z caron
Ç	\(,C*	\[Ccedilla]	\U'00C7'	C cedilla
ç	\(,c*	\[ccedilla]	\U'00E7'	c cedilla
Å	\(oA*	\[Aring]	\U'00C5'	A ring
å	\(oa*	\[aring]	\U'00E5'	a ring
˝	\(a"*	\[hungarumlaut]	\U'02DD'	Hungarian umlaut
¯	\(a-*	\[macron]	\U'00AF'	overbar accent
˙	\(a.*	\[dotaccent]	\U'02D9'	dot accent
ˆ	\(a^*	\[circumflex]	\U'005E'	circumflex accent
´	\(aa	\[acute]	\U'00B4'	acute accent
`	\(ga	\[grave]	\U'0060'	grave accent
˘	\(ab*	\[breve]	\U'02D8'	breve accent
¸	\(ac*	\[cedilla]	\U'00B8'	cedilla accent
¨	\(ad*	\[dieresis]	\U'00A8'	umlaut accent
ˇ	\(ah*	\[caron]	\U'02C7'	caron accent
˚	\(ao*	\[ring]	\U'02DA'	ring accent
˜	\(a~*	\[tilde]	\U'007E'	tilde accent
˛	\(ho*	\[ogonek]	\U'02dB'	hook accent
^	\(ha*	\[asciicircum]	\U'005E'	high circumflex
~	\(ti*	\[asciitilde]	\U'007E'	tilde in the vertical middle
„	\(Bq*	\[quotedblbase]	\U'201E'	low double comma quote
‚	\(bq*	\[quotesinglbase]	\U'201A'	low single comma quote
“	\(lq	\[quotedblleft]	\U'201C'	left double quote
“	\(``			*(alias)*
”	\(rq	\[quotedblright]	\U'201D'	right double quote
”	\(''			*(alias)*
‘	\(oq	\[quoteleft]	\U'2018'	single open quote
’	\(cq	\[quoteright]	\U'2019'	single closing quote
'	\(aq	\[quotesingle]	\U'0027'	apostrophe quote
"	\(dq	\[quotedbl]	\U'0022'	double quote
«	\(Fo*	\[guillemotleft]	\U'00AB'	left guillemet
»	\(Fc*	\[guillemotright]	\U'00BB'	right guillemet
‹	\(fo*	\[guilsinglleft]	\U'2039'	single left angle quotation mark
›	\(fc*	\[guilsinglright]	\U'203A'	single right angle quotation mark

Output	*Input*	*PostScript*	*Unicode*	*Notes*
¡	\(r!*	\[exclamdown]	\U'00A1'	inverted exclamation mark
¿	\(r?*	\[questiondown]	\U'00BF'	inverted question mark
—	\(em	\[emdash]	\U'2014'	em-dash symbol
–	\(en	\[endash]	\U'2013'	en-dash symbol
‐	\(hy	\[hyphen]	\U'2010'	hyphen symbol
[	\(lB*	\[bracketleft]	\U'005B'	left square bracket
]	\(rB*	\[bracketright]	\U'005D'	right square bracket
{	\(lC*	\[braceleft]	\U'007B'	left curly bracket
}	\(rC*	\[braceright]	\U'007D'	right curly bracket
⟨	\(la*	\[angleleft]	\U'27E8'	mathematical left angle bracket
⟩	\(ra*	\[angleright]	\U'27E9'	mathematical right angle bracket
⎪	\(bv	\[braceex]	\U'23AA'	curly brace vertical extension
⎡		\[bracketlefttp]	\U'23A1'	left square bracket top
⎣		\[bracketleftbt]	\U'23A3'	left square bracket bottom
⎢		\[bracketleftex]	\U'23A2'	left square bracket extension
⎤		\[bracketrighttp]	\U'23A4'	right square bracket top
⎦		\[bracketrightbt]	\U'23A6'	right square bracket bottom
⎥		\[bracketrightex]	\U'23A5'	right square bracket extension
⎧	\(lt	\[bracelefttp]	\U'23A7'	left curly brace top
⎨	\(lk	\[braceleftmid]	\U'23A8'	left curly brace middle
⎩	\(lb	\[braceleftbt]	\U'23A9'	left curly brace bottom
		\[braceleftex]	\U'23AA'	left curly brace extension
⎫	\(rt	\[bracerighttp]	\U'23AB'	right curly brace top
⎬	\(rk	\[bracerightmid]	\U'23AC'	right curly brace middle
⎭	\(rb	\[bracerightbt]	\U'23AD'	right curly brace bottom
		\[bracerightex]	\U'23AA'	right curly brace extension
⎛		\[parenlefttp]	\U'239B'	left parenthesis top
⎝		\[parenleftbt]	\U'239D'	left parenthesis bottom
⎜		\[parenleftex]	\U'239C'	left parenthesis extension
⎞		\[parenrighttp]	\U'239E'	right parenthesis top
⎠		\[parenrightbt]	\U'23A0'	right parenthesis bottom
⎟		\[parenrightex]	\U'239F'	right parenthesis extension
←	\(<-	\[arrowleft]	\U'2190'	arrow left
→	\(->	\[arrowright]	\U'2192'	arrow right
↔	\(<>*	\[arrowboth]	\U'2194'	horizontal arrow in both directions
↓	\(da	\[arrowdown]	\U'2193'	arrow down
↑	\(ua	\[arrowup]	\U'2191'	arrow up
	\(va*	\[arrowupdn]	\U'2195'	vertical arrow in both directions
⇐	\(lA*	\[arrowdblleft]	\U'21D0'	double arrow left
⇒	\(rA*	\[arrowdblright]	\U'21D2'	double arrow right
⇔	\(hA*	\[arrowdblboth]	\U'21D4'	horizontal double arrow in both directions
⇓	\(dA*	\[arrowdbldown]	\U'21D3'	double arrow down
⇑	\(uA*	\[arrowdblup]	\U'21D1'	double arrow up
	\(vA*	\[uni21D5]	\U'21D5'	vertical double arrow in both directions
⎯	\(an*	\[arrowhorizex]	\U'23AF'	horizontal arrow extension
|	\(ba*	\[bar]	\U'007C'	bar
│	\(br	\[SF110000]	\U'2502'	box rule
_	\(ul	\[underscore]	\U'005F'	underscore
	\(rn	\[overline]	\U'203E'	overline
_	\(ru			baseline rule
¦	\(bb*	\[brokenbar]	\U'00A6'	broken bar

Output	*Input*	*PostScript*	*Unicode*	*Notes*
⁄	\(sl	\[slash]	\U'002F'	slash
\	\(rs	\[backslash]	\U'005C'	revers slash
○	\(ci	\[circle]	\U'25CB'	circle
•	\(bu	\[bullet]	\U'2022'	bullet
‡	\(dd	\[daggerdbl]	\U'2021'	double dagger
†	\(dg	\[dagger]	\U'2020'	dagger
◊	\(lz*	\[lozenge]	\U'25CA'	diamond
□	\(sq	\[uni25A1]	\U'25A1'	square
	\(ps*	\[paragraph]	\U'00B6'	end of paragraph marker
§	\(sc	\[section]	\U'00A7'	section sign
☜	\(lh	\[uni261C]	\U'261C'	hand pointing left
☞	\(rh	\[a14]	\U'261E'	hand pointing right
@	\(at	\[at]	\U'0040'	at
#	\(sh*	\[numbersign]	\U'0023'	number sign
↵	\(CR*	\[carriagereturn]	\U'21B5'	carriage return
	\(OK*	\[a19]	\U'2713'	check mark
©	\(co	\[copyright]	\U'00A9'	copyright sign
®	\(rg	\[registered]	\U'00AE'	registered mark
™	\(tm	\[trademark]	\U'2122'	trademark symbol
	\(bs			AT&T Bell Labs logo
$	\(Do*	\[dollar]	\U'0024'	dollar
¢	\(ct	\[cent]	\U'00A2'	cent
€	\(Eu*	\[Euro]	\U'20AC'	font-specific Euro glyph
¥	\(Ye*	\[yen]	\U'00A5'	Japanese Yen
£	\(Po*	\[sterling]	\U'00A3'	pound sterling
¤	\(Cs*	\[currency]	\U'00A4'	Scandinavian currency sign
ƒ	\(Fn*	\[florin]	\U'0192'	Dutch currency sign
°	\(de	\[degree]	\U'00B0'	degree
‰	\(%0*	\[perthousand]	\U'2030'	per thousand
′	\(fm	\[minute]	\U'2032'	arc minute sign
″	\(sd*	\[second]	\U'2033'	arc second sign
µ	\(mc*	\[mu]	\U'00B5'	micro sign
ª	\(Of*	\[ordfeminine]	\U'00AA'	feminine ordinal
º	\(Om*	\[ordmasculine]	\U'00BA'	masculine ordinal
∧	\(AN*	\[logicaland]	\U'2227'	logical and
∨	\(OR*	\[logicalor]	\U'2228'	logical or
¬	\(no			logical not
¬	\[tno]*	\[logicalnot]	\U'00AC'	text variant of 'no'
∃	\(te	\[existential]	\U'2203'	there exists
∀	\(fa	\[universal]	\U'2200'	for all
∋	\(st*	\[suchthat]	\U'220B'	such that
∴	\(3d*	\[therefore]	\U'2234'	therefor
∴	\(tf*			*(alias)*
|	\(or	\[bar]	\U'007C'	bitwise OR
½	\(12	\[onehalf]	\U'00BD'	1/2 symbol
¼	\(14	\[onequarter]	\U'00BC'	1/4 symbol
¾	\(34	\[threequarters]	\U'00BE'	3/4 symbol
	\(18*	\[oneeighth]	\U'215B'	1/8 symbol
	\(38*	\[threeeighths]	\U'215C'	3/8 symbol
	\(58*	\[fiveeighths]	\U'215D'	5/8 symbol

Output	*Input*	*PostScript*	*Unicode*	*Notes*
	\(78*	\[seveneighths]	\U'215E'	7/8 symbol
^{1}	\(S1*	\[onesuperior]	\U'00B9'	superscript 1
^{2}	\(S2*	\[twosuperior]	\U'00B2'	superscript 2
^{3}	\(S3*	\[threesuperior]	\U'00B3'	superscript 2
+	\(pl	\[plus]	\U'002B'	plus
−	\(mi	\[minus]	\U'2212'	minus
	\(−+*	\[uni2213]	\U'2213'	minus-plus
±	\(+−			plus-minus
±	\[t+−]*	\[plusminus]	\U'00B1'	text variant of plus-minus
·	\(pc*	\[periodcentered]	\U'00B7'	period centered
⋅	\(md*	\[dotmath]	\U'22C5'	multiplication dot
×	\(mu			multiply sign
×	\[tmu]*	\[multiply]	\U'00D7'	text variant of multiply sign
⊗	\(c**	\[circlemultiply]	\U'2297'	multiply sign in circle
⊕	\(c+*	\[circleplus]	\U'2295'	plus sign in circle
÷	\(di			division sign
÷	\[tdi]*	\[divide]	\U'00F7'	text variant of division sign
⁄	\(f/*	\[fraction]	\U'2044'	bar for fractions
∗	\(**	\[asteriskmath]	\U'2217'	mathematical asterisk
≤	\(<=	\[lessequal]	\U'2264'	less or equal
≥	\(>=	\[greaterequal]	\U'2265'	greater or equal
	\(<<*	\[uni226A]	\U'226A'	much less
	\(>>*	\[uni226B]	\U'226B'	much greater
=	\(eq	\[equal]	\U'003D'	equal
≠	\(!=	\[notequal]	\U'2260'	not equal
≡	\(==	\[equivalence]	\U'2261'	equivalent
	\(ne*	\[uni2262]	\U'2262'	not equivalent
≅	\(=~*	\[congruent]	\U'2245'	congruent
~	\(ap	\[similar]	\U'223C'	similar
≈	\(~~	\[approxequal]	\U'2248'	almost equal to
∝	\(pt	\[proportional]	\U'221D'	proportional
∅	\(es	\[emptyset]	\U'2205'	empty set
∈	\(mo	\[element]	\U'2208'	element of
∉	\(nm*	\[notelement]	\U'2209'	not element of
⊂	\(sb	\[propersubset]	\U'2282'	proper subset
⊄	\(nb*	\[notsubset]	\U'2284'	not subset
⊃	\(sp	\[propersuperset]	\U'2283'	proper superset
	\(nc*	\[uni2285]	\U'2285'	not superset
⊆	\(ib	\[reflexsubset]	\U'2286'	subset or equal
⊇	\(ip	\[reflexsuperset]	\U'2287'	superset or equal
∩	\(ca	\[intersection]	\U'2229'	intersection
∪	\(cu	\[union]	\U'222A'	union
∠	\(/_*	\[angle]	\U'2220'	angle
	\(pp*	\[perpendicular]	\U'22A5'	perpendicular
∫	\(is	\[integral]	\U'222B'	integral
Σ	\(*S	\[summation]	\U'2211'	summation
∑	\[sum]*			*(alias)*
Π	\(*P	\[product]	\U'220F'	product
		\[coproduct]*	\U'2210'	coproduct
∇	\(gr	\[gradient]	\U'2207'	gradient
√	\(sr	\[radical]	\U'221A'	square root

Output	*Input*	*PostScript*	*Unicode*	*Notes*
√		`\[sqrt]`*		*(alias)*
‾		`\[radicalex]`		square root continuation
		`\[sqrtex]`*		*(alias)*
⌈	`\(lc`	`\[uni2308]`	`\U'2308'`	left ceiling
⌉	`\(rc`	`\[uni2309]`	`\U'2309'`	right ceiling
⌊	`\(lf`	`\[uni230A]`	`\U'230A'`	left floor
⌋	`\(rf`	`\[uni230B]`	`\U'230B'`	right floor
∞	`\(if`	`\[infinity]`	`\U'221E'`	infinity
ℵ	`\(Ah`*	`\[aleph]`	`\U'2135'`	aleph
ℑ	`\(Im`*	`\[Ifraktur]`	`\U'2111'`	Gothic I, imaginary
ℜ	`\(Re`*	`\[Rfraktur]`	`\U'211C'`	Gothic R, real
℘	`\(wp`*	`\[weierstrass]`	`\U'2118'`	Weierstrass p
∂	`\(pd`	`\[partialdiff]`	`\U'2202'`	partial differentiation
	`\(-h`*	`\[hbar]`	`\U'210F'`	Planck constant / 2pi (h-bar)
ϑ	`\(+h`*	`\[theta1]`	`\U'03D1'`	variant theta
φ	`\(+f`*	`\[phi1]`	`\U'03C6'`	variant phi
ϖ	`\(+p`*	`\[omega1]`	`\U'03D6'`	variant pi
	`\(+e`*	`\[uni03F5]`	`\U'03F5'`	variant epsilon
♣	`\(CL`*	`\[club]`	`\U'2663'`	black club suit
♠	`\(SP`*	`\[spade]`	`\U'2660'`	black spade suit
♥	`\(HE`*	`\[heart]`	`\U'2665'`	black heart suit
		`\[uni2661]`	`\U'2661'`	white heart suit
♦	`\(DI`*	`\[diamond]`	`\U'2666'`	black diamond suit
		`\[uni2662]`	`\U'2662'`	white diamond suit