

## NROFF Users' Manual

*Joseph F. Ossanna  
Bell Laboratories, Murray Hill, New Jersey*

### Introduction

NROFF is a text processor on the PDP-11/45 UNIX Time-Sharing System. It accepts lines of text interspersed with lines of format control information (request lines) and formats the text into a printable, paginated document having a user-designed style. The request syntax and many of the requests themselves are reminiscent of other text formatters\*. NROFF differs from its predecessors in offering unusual freedom in document styling. Examples include: arbitrary style headers and footers; arbitrary style footnotes; multiple automatic sequence numbering for paragraphs, sections, etc; and multiple column output (with the aid of a post-processor†). TROFF, a similar text processor that produces output for a Graphic Systems Phototypesetter, is also available on UNIX<sup>1</sup>.

### Usage

The general form of invoking NROFF at UNIX command level is

`nroff options files`

where "options" represents any of a number of optional arguments and "files" represents the list of files containing the document to be formatted. An argument consisting of a single minus "-" sign is taken to be a file name corresponding to the standard input. The options, which may appear in any order so long as they appear before the files, are:

#### *Option Effect*

- +N** Commence output at the first page whose page number is N (independent of whether or not the page number is being printed).
- N** End output *after* the first page whose page number is N.
- nN** Number first generated page N; simulates an initial ".pn N" request.
- s** Stop between pages. Printing will halt prior to each page (including the first) to permit paper loading or changing. Printing is resumed by typing a "new-line".
- h** High speed output. Strings of space characters are replaced by tab characters where possible to speed printing and to reduce the number of output characters.
- i** Read standard input after the input files are exhausted. NROFF automatically reads the standard input, if no input files are given.
- mx** Simulates an ".so /usr/lib/tmac.x" request.

---

\*Particularly ROFF on the same System and M. D. McIlroy's ROFF on the Murray Hill Computation Center HIS-6070.

†See description of "COL" in the UNIX Programmer's Manual and tutorial part of this manual.

- q The prompt names for insertions are not printed and the bell character is sent instead; in addition, the insertion is not echoed. This mode permits insertions during the actual output printing (See description of the "rd" request).

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

```
nroff +7 -h -s file1 file2
```

requests formatting of a document contained in the files named "file1" and "file2", beginning with page 7, with high-speed output, and stopping before every page.

If no files are given, the input is taken from the standard input, which may be a "pipe"<sup>2</sup>. For example, NROFF may be used with NEQN<sup>3</sup>, an equation preprocessor, as follows:

```
neqn files | nroff options
```

The "|" indicates the piping of NEQN's output to NROFF's input.

The remainder of this manual consists of: a Request Summary and Index; a Reference Manual keyed to the index; and a set of Tutorial Examples.

## References

- [1] J. F. Ossanna, *TROFF User's Manual*, Internal Document, April 1974.
- [2] K. Thompson, D. M. Ritchie, *UNIX Programmer's Manual*, Fifth Edition (June 1974).
- [3] B. W. Kernighan, L. L. Cherry, *Typesetting Mathematics – User's Guide*, Internal Document, March 1974.

## REQUEST SUMMARY AND INDEX

<i>Request Form</i>	<i>Initial Value</i>	<i>If no Argument</i>	<i>Cause Break*</i>	<i>Explanation</i>
<b>I. Page Control</b>				
.pl ±N	66 lines	66 lines	no	Page length.
.bp ±N	N=1	-	yes	Begin page.
.pn ±N	N=1	ignored	no	Page number.
.po ±N	N=0	prev	no	Page offset.
.ne N	-	N=1	no	Need N lines.
.mk a	line 0	internal	no	Mark current line.
.rt -N	-	internal	no	Return to marked line.
<b>II. Text Filling, Adjusting, and Centering</b>				
.br	-	-	yes	Break.
.fi	fill	-	yes	Fill output lines.
.nf	fill	-	yes	No filling and adjusting.
.ad c	adj,norm	adjust	no	Adjust mode on.
.na	adjust	-	no	No adjusting.
.ce N	off	N=1	yes	Center N input text lines.
<b>III. Vertical Spacing</b>				
.ls N	N=1	prev	no	Line spacing.
.sp N	-	N=1	yes	Space N lines.
.lv N	-	N=1	no	See "sv" below.
.sv N	-	N=1	no	Save N lines.
.os	-	-	no	Output saved lines.
.ns	space	-	no	No-space mode on.
.rs	-	-	no	Restore spacing.
.xh	off	-	no	Extra-half-line mode on.
<b>IV. Line Length and Indenting</b>				
.ll ±N	65	prev	no	Line length.
.in ±N	N=0	prev	yes	Indent.
.ti ±N	-	N=1	yes	Temporary indent.
<b>V. Macros, Diversion, and Line Traps</b>				
.de xx	-	ignored	no	Define or redefine a macro.
.am xx	-	ignored	no	Append to a macro.
.ds xx	-	ignored	no	Define a string.
.as xx	-	ignored	no	Append to a string.
.rm xx	-	ignored	no	Remove macro or string.
.di xx	-	end	no	Divert output to macro "xx".
.da xx	-	end	no	Divert and append to "xx".
.wh -N xx	-	-	no	When; set a line trap.
.ch -N -M	-	-	no	Change trap location.
.ch xx -M	-	-	no	"
.em xx	none	none	no	End-macro name specification.

---

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

<i>Request Form</i>	<i>Initial Value</i>	<i>If no Argument</i>	<i>Cause Break</i>	<i>Explanation</i>
---------------------	----------------------	-----------------------	--------------------	--------------------

#### VI. Number Registers

.nr a ±N –M	-		no	Number register.
.nr ab ±N –M	-		no	"
.nc c \n	\n	\n	no	Number character.
.ar	arabic	-	no	Arabic numbers.
.ro	arabic	-	no	Lower case roman numbers.
.RO	arabic	-	no	Upper case roman numbers.

#### VII. Tabs, Leaders, and Fields

.ta N,...	9,17,...	none	no	Tab settings.
.tc c	unpaddable space	unpaddable space	no	Tab replacement character
.lc c	period	period	no	Leader replacement character.
.fc a b	off	off	no	Set field delimiter and pad characters.

#### VIII. Input and Output Conventions and Character Translations

.ec c \	\		no	Set escape character.
.cc c ;	;		no	Basic control character.
.c2 c			no	Nobreak control character.
.li N -		N=1	no	Accept input lines literally.
.tr abcd....	none	-	no	Translate on output.

#### IX. Local Horizontal and Vertical Motions

.ul N -		N=1	no	Underline input text lines.
---------	--	-----	----	-----------------------------

#### X. Hyphenation.

.nh	hyphenate	-	no	No hyphenation.
.hy	hyphenate	-	no	Hyphenate.
.hc c	none	none	no	Hyphenation indicator character.

#### XI. Three Part Titles.

.tl 'left'center'right'	-		no	Title.
.lt N 65		prev	no	Length of title.

#### XII. Output Line Numbering.

.nm ±N M S I		off	no	Number mode on or off, set parameters.
.np M S I	none	reset	no	Number parameters set or reset.

#### XIII. Conditional Input Line Acceptance

.if c anything	-		no	If condition true accept "anything".
.if !c anything	-		no	"
.if N anything	-		no	"
.if !N anything	-		no	"

#### XIV. Environment Switching.

.ev N N=0		prev	no	Environment <i>pushed down</i> (10 levels).
-----------	--	------	----	---

#### XV. Insertions from the Standard Input Stream

.rd prompt -		bell	no	Read insert.
.ex -		-	no	Exit.

<i>Request Form</i>	<i>Initial Value</i>	<i>If no Argument</i>	<i>Cause Break</i>	<i>Explanation</i>
---------------------	----------------------	-----------------------	--------------------	--------------------

#### XVI. Input File Switching

.so filename	-		no	Switch source file ( <i>pushed down</i> ; 5 levels).
.nx filename	-		no	Next file.

#### XVII. Miscellaneous

.tm string	-	-	no	Teletype message.
.ig	-	-	no	Ignore.
.fl	-	-	yes	Flush output buffer.
.ab	-	-	no	Abort.

#### XVIII. Error Messages

#### Escape Sequences for Characters, Indicators, and Functions

<i>Ref.</i>	<i>Input</i>	<i>Meaning</i>
-------------	--------------	----------------

-	\\	\ (to prevent or delay the interpretation of \)
-	\e	Directly printable version of the current escape character.
II	\(space)	Unpaddable space character
VIII.IV	\&	Non-printing, zero width character
VIII.VII	\!	Transparent line indicator
VIII.VIII	\?	Raw transmission indicator
V.III	\\$	Argument indicator
V.I	\*	String indicator
-	\:	Generates ASCII ETX (003) for post processor use
VII.I	\a	Non-interpreted leader character
VIII.II	\c	Interrupt text processing
IX.I	\d	Forward (down) 1/2 Line vertical motion
-	\f	Font change function (for TROFF compatibility)
IX.III	\k	Mark horizontal input place
-	\l	ASCII Delete
VI	\n	Number register indicator
II	\p	Break and spread output line
IX.I	\r	Reverse 1 Line vertical motion
-	\s	Point-size change function (for TROFF compatibility)
VII.I	\t	Non-interpreted horizontal tab
IX.I	\u	Reverse (up) 1/2 Line vertical motion
-	\x	ASCII Shift-out
-	\y	ASCII Shift-in
-	\(newline)	Concealed (ignored) newline

## REFERENCE MANUAL

### Explanation

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

*Numerical parameter input.* Numerical arguments are indicated below in several symbolic forms:  $\pm N$  means that the argument may take the forms  $N$ ,  $+N$ , or  $-N$  and that the corresponding effect is to set the affected parameter to  $N$ , to increment it by  $N$ , or decrement it by  $N$  respectively;  $-N$  means that the argument may take the form  $N$  or  $-N$  and that the effect is to set the parameter to  $N$  or  $-N$ . Other capital letters are used for additional numerical arguments. When number arithmetic is used in the  $\pm N$  case, the entire  $N$  is evaluated before the result is taken as an increment (e. g.  $-11-6$  will decrement by  $(11-6)$ ). Certain requests that set a parameter value will restore the previously set value, if no new value is specified; only the most recent previous value is saved unless otherwise noted.

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

### I. Page control

Top and bottom margins are not automatically provided; it is conventional to define two macros and to set traps for them at vertical positions 0 (top) and  $-N$  (from the bottom). See § V and Tutorial Examples.

<i>Request Form</i>	<i>Initial Value</i>	<i>If no Argument</i>	<i>Cause Break</i>	<i>Explanation</i>
.pl $\pm N$	66 lines	66 lines	no	Page length set to $\pm N$ lines.
.bp $\pm N$	$N=1$	-	yes	Begin page. The current page is ejected and a new page is begun. If $\pm N$ is given, the new page number will be $\pm N$ . See request "ns".
.pn $\pm N$	$N=1$	ignored	no	Page number. The next page (when it occurs) will have the page number $\pm N$ . A "pn" occurring <i>before</i> the first break or first text will set the page number for the first page.
.po $\pm N$	$N=0$	prev	no	Page offset. $\pm N$ spaces are prepended to each output line; i. e. the page image is moved $\pm N$ spaces to the right.
.ne $N$	-	$N=1$	no	Need $N$ lines. If the distance, $D$ , to the next trap position is less than $N$ lines, the paper is moved forward the $D$ lines, which will spring the trap. If there are no remaining traps on the page, $D$ will be the distance to the bottom of the page. See § V.
.mk a	line 0	internal	no	Mark current line. The current number of lines on the page is stored for future reference. If "a" is given, the number is stored in number register "a". (see "rt" request.)
.rt $-N$	-	internal	no	Return to marked line. The paper will be moved in the <i>reverse direction only</i> to a place a $N$ lines from the top of the page or, if $N$ is negative, to a place a $N$ lines before the current line, or, if $N$ is absent, to a previously marked line (see request "mk"), if any.

## II. Text Filling, Adjusting, and Centering

The default fill mode is to fill output lines; input words are taken from the next input line or output words are deferred until the next output line to produce output lines that are full but within the current line length size. a "word" is a string of characters separated by spaces. If two words must be separated by spaces that must not be used for adjustment or line splitting, the unpaddable space character "\ " may be used or the "tr" request may be used (§VIII.VI). The default adjust mode is to adjust lines for a uniform right (as well as left) margin; if a fully formed line contains fewer character positions than the current line length, the blank spaces between words are expanded to achieve the current line length. Both of these processes may be turned off. During filling hyphenation is automatically attempted when the next word does not fit on the line; this process may be turned off also. (See §X).

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.

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

## III. Vertical Spacing

*I. Line spacing.* The default line spacing is single space. The line spacing may be dynamically set and reset with the "ls" request. If the line spacing is N, N-1 blank lines are put out *after* each output line. If a trap occurs, any remaining blank lines are omitted.

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

<i>Request Form</i>	<i>Initial Value</i>	<i>If no Argument</i>	<i>Cause Break</i>	<i>Explanation</i>
.ls N	N=1	prev	no	Line spacing set to $\pm N$ . If N, N-1 blank lines are interpolated between output lines.
.sp N	-	N=1	yes	Space N lines in either direction. If N is negative, the motion is upward (backward). Downward (forward) motion is limited to the distance to the nearest trap. Reverse motion is limited to the distance to the top of the page. If the no-space mode is on, no spacing occurs.
.lv N	-	N=1	no	See "sv" request next.
.sv N	-	N=1	no	Save N lines of space. If the distance to the next trap is greater than N, N lines is output. No-space mode has no effect. If this distance is less than N, no lines is immediately output, but N is remembered for later output (see request "os"). Subsequent "sv" requests will overwrite any still remembered N.
.os	-	-	no	Output saved blank lines. No-space mode has no effect. Used to finally output a block of lines requested by the "sv" request.
.ns	space	-	no	No-space mode turned on. When on, the no-space mode inhibits "sp" requests and "bp" requests without a next page number. The no-space mode is turned off when a line of output is produced.
.rs	space	-	no	Restore spacing. The no-space mode is turned off.
Blank text line.	-	-	yes	Causes a break and output of a blank line exactly like "sp 1".
.xh	off	-	no	Extra-half-line mode is set on. When this mode is on each output line has an additional half-line-forward control sequence (escape-9) appended. If single spacing is in effect, the net effect is to cause 1.5-line spacing. NROFF is otherwise unaware of this mode so that it is necessary for the user to set the page length, margins, etc. to 2/3 of the actual vertical white space wanted.

#### IV. Line Length and Indenting

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

<i>Request Form</i>	<i>Initial Value</i>	<i>If no Argument</i>	<i>Cause Break</i>	<i>Explanation</i>
.ll $\pm N$	65	prev	no	Line Length is set to $\pm N$ .
.in $\pm N$	N=0	prev	yes	Indent is set to $\pm N$ . If the line length is L and the indent is N, N spaces are put out at the beginning of each output line and the text on the remainder of the line is constrained to a size L-N.
.ti $\pm N$	-	N=1	yes	Temporary indent. The next output text line will be indented $\pm N$ spaces ( $\pm N$ with respect to the current indent). The resulting temporary indent <i>may not</i> be negative. The current indent is not changed.



## V. Macros, Strings, Diversion, and Line Traps

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

*II. Copy mode input interpretation.* During macro and string definition (or appending to) the input is processed in a "copy" mode. The only input processing that occurs is escape mapping (see § VIII./) and: (1) number registers indicated by "\n" are expanded; (2) string references (indicated by "\\*") are interpolated into the input; and (3) argument references (indicated by "\\$") are interpolated into the input. In cases (1-3) the interpretation can be suppressed by prepending a "\". Since "\\" maps into a "\", "\\n" will copy as "\n" which will be interpreted as a number register indicator when the macro or string is reread.

The concealed newline (\(newline)) is always processed (thrown away). Tab, leader, and field-mechanism characters are not processed in copy mode. Except for the interpretation of number registers, strings, and arguments, functions are not interpreted or performed.

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

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

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

```
\$N
```

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

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

and called by

```
.xx Monday 14 th
```

to produce the text

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

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

<i>Request Form</i>	<i>Initial Value</i>	<i>If no Argument</i>	<i>Cause Break</i>	<i>Explanation</i>
.de xx	-	ignored	no	Define or redefine a macro with the one or two character name "xx". The contents of the macro begins on the next input line. Input lines are copied until the definition is terminated by a line beginning with "..". A macro may contain "de" requests provided the corresponding ".." is concealed to prevent copy termination; "\\.." will copy as "\\.." and be reread as "..".
.am xx	-	-	no	Append to macro (append version of "de").
.ds xx string			-	no Define string. The character string "string" (without a terminal new-line character) is defined as a macro having the (one or two character) name "xx". An initial double-quote, "\"", is stripped off to permit initial blanks. This request is like the "de" request, except that the resulting macro consists of only "string".
.as xx string			-	no Append to string (append version of "ds").
.rm xx	-	-	no	Remove macro or string. The macro or string named "xx" is removed from the name list. Subsequent references will have no effect.
.di xx	-	end	no	Divert output into the macro named "xx". The macro name "xx" is (re)defined at this point. Normal text processing occurs during diversion except that page offsetting is not done. The diversion ends when the request "di" or "da" is encountered without an argument. Diversions cannot be nested. The vertical size of the diverted output is kept in a number register (named "dn") for possible later use.
.da xx	-	end	no	Append to diversion (append version of "di").
.wh -N xx	-	-	no	When (after) output line -N occurs invoke macro "xx". Any macro previously planted at line -N is replaced by "xx". A positive N is measured from the top of the page and -N is measured from the bottom. If the page length is 66, line -1 is line 66. A zero N will set a trap at the beginning of a page and should be used for headers. If no macro name is given, the trap planted at line -N, if any, is removed.
.ch -N -M -	-	-	no	See next.
.ch xx -M -	-	-	no	Change the trap planted at line -N to occur instead at line -M. Alternatively, change the place at which the macro "xx" is planted to be -M. If no trap exists at line -N, the request is ignored.

## VI. Number Registers

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

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

where "\n" is the "number character" indicating that the next character(s), unless "+" or "(", is the name of a number register and where "a" or "ab" is the number register name. The "+" in the second

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

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

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

<i>Name</i>	<i>Use</i>
%	Current page number.
yr	Last two digits of current year.
mo	Current month (1-12).
dy	Current day of the month.
dw	Current day of the week (1-7).
nl	Current number of output lines on a page.
dn	Size in lines of last diversion.
hp	Current number of character positions read on input line.
.v	Current vertical line spacing (read only).
.p	Current page length (read only).
.o	Current page offset (read only)
.l	Current line length (read only).
.i	Current indent (read only).
.t	Distance to the next trap (read only).
.\$	Number of arguments available at the current macro level (read only).
.x	Reserved version-dependent register (read only).

## VII. Tabs, Leaders, and Fields

*1. Tabs and leaders.* The horizontal tab character is replaced by a number of unpaddable space characters corresponding to the distance to the next tab stop on that *input* line. The unpaddable space character which may be otherwise input by "\ " (\space) prints as a space but is not recognized as a space meaning word separation. Optionally, the tab may be replaced by a user-specified replacement-character string having enough replacement-characters to fill out the distance to the tab stop. The "leader" character (ASCII "SOH") is treated exactly like the horizontal tab except that the default mode is to use "." as the replacement-character. Tabs or leaders encountered after the last stop generate a single replacement character.

*II. Fields.* A "field" is surrounded by a user-defined field delimiter character. A field contains a string consisting of sub-strings separated by padding indicator characters. The field length is the distance on the *input* line from the place where the field begins to the next tab stop. The difference between the total length of all the sub-strings and the field length is incorporated as padding space that is divided among the indicated padding places. The incorporated padding consists of unpaddable spaces or, if necessary, of backspaces. For example, if the field delimiter is "/", and if the padding indicator is "^", /xxx/ and /^xxx^/ represent right-adjusted and centered "xxx" respectively.

<i>Request Form</i>	<i>Initial Value</i>	<i>If no Argument</i>	<i>Cause Break</i>	<i>Explanation</i>
.ta N,...	9,17,25,...	none	no	Tab settings. Default tabs are at 9, 17, 25, ..., 160 (each worth 8 spaces); a total of 20 stops may be set. The stop values may be separated by commas, spaces, or any other nonnumerics.
.tc c	unpaddable space	unpaddable space	no	The tab replacement-character is set to "c" or reset to unpaddable space.
.lc c	. (period)	. (period)	no	Leader replacement character is set to "c" or reset to period.
.fc a b	off	off	no	The field delimiter is set to "a"; the padding indicator character is set to "space" or to "b", if given. In the absence of arguments the field mechanism is turned off.

### VIII. Input and-Output Conventions and Character Translations

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

*I. Input character translations.* The ASCII controls "SOH", bell, backspace, tab, newline, shift-in, shift-out, and escape are accepted and *all* others are ignored. The character "\" plays an important role during input by modifying the interpretation of the character following; for this reason the "\" is called the "escape" character but it should not be confused with the ASCII control character of the same name. For example, "\" generates an unpaddable rather than an ordinary space character. In addition, the "\" is used to introduce various indicators, functions, and local motions (see later). A complete list of characters modified by a preceding "\" follows; the treatment of all other characters is unaffected.

#### *Ref. Input Meaning*

-	\\	\ (to prevent or delay the interpretation of \)
-	\e	Directly printable version of the current escape character.
II	\(space)	Unpaddable space character
VIII.IV	\&	Non-printing, zero width character
VIII.VII	\!	Transparent line indicator
VIII.VIII	\?	Raw transmission indicator
V.III	\\$	Argument indicator
V.I	\*	String indicator
-	\:	Generates ASCII ETX (003) for post processor use
VII.I	\a	Non-interpreted leader character
VIII.II	\c	Interrupt text processing
IX.I	\d	Forward (down) 1/2 Line vertical motion
-	\f	Font change function (for TROFF compatibility)
IX.III	\k	Mark horizontal input place
-	\l	ASCII Delete
VI	\n	Number register indicator
II	\p	Break and spread output line
IX.I	\r	Reverse 1 Line vertical motion
-	\s	Point-size change function (for TROFF compatibility)
VII.I	\t	Non-interpreted horizontal tab

IX./     \u Reverse (up) 1/2 Line vertical motion  
-        \x ASCII Shift-out  
-        \y ASCII Shift-in  
-        \ (newline) Concealed (ignored) newline

The escape character may be changed.

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

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

*III. Backspacing, underlining, overstriking, etc.* Underlining and generalized overstriking are discussed in §IX.II).

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

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

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

*VI. Output translation.* Provision exists for specifying a mapping of any character into any other character. All text processing (e. g. character comparisons) takes place with the original character which appears to have the width of the final character. The graphic translation occurs at the moment of output (including diversion).

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.tr abcd....	none	-	no	Translate. "a" will be mapped into "b", "c" will be mapped into "d", etc. If an odd number of characters is given, the last one will be mapped into the space character.

A common use of the "tr" request is to provide nonadjustable spaces. When normal filling and adjusting is done the space character between words indicates where the additional space may be put and where line splitting may occur. A ".tr |", which specifies that "|" be mapped into a space during output,

permits tying two or more words together so that they will neither be moved apart nor split across two lines. Examples might be "Fig.|12", "Mr.|Smith", and "2|+|x|=|y". Another way to get nonadjustable spaces is to use unpaddable space character "\ ". It should be noted that horizontal tabs are converted into unpaddable space characters.

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

*VIII. Raw Transmission.* An input line beginning with "\" initiates a raw transmission mode in which all input bytes (except null and the one selected to mean end of raw mode) are passed through NROFF untouched and without interpretation. The byte meaning end of raw transmission is defined as the first byte following the "\"; this byte is not transmitted.

## IX. Local Horizontal and Vertical Motions

*I. Local Vertical Motions.* The escape sequences \u, \d, and \r may be used to obtain the local vertical motions 1/2 Line up, 1/2 Line down, and 1 Line up respectively. They are equivalent to the input sequences "(ESC)8", "(ESC)9", and "(ESC)7" respectively and where "(ESC)" represents the ASCII "escape" character. For example, "E<sub>2</sub>" could be generated by the sequence "E\d2\u" or "E(ESC)92(ESC)8".

*II. Local Horizontal Motions.* Underlining and other overstriking can be achieved by backspacing and overstriking with the desired character(s). Because words separated by spaces are the entities being processed, it is unwise to backspace over a space that may later be enlarged by line adjustment or even taken as a good place to end an output line. Because underlining is common, provision exists for automatic underlining of input lines.

<i>Request Form</i>	<i>Initial Value</i>	<i>If no Argument</i>	<i>Cause Break</i>	<i>Explanation</i>
.ul N	-	N=1	no	UNDERline the next N input text lines. Only alphanumeric characters are underlined. If N > 1 is used, it must be realized that macros invoked by line traps may be interpolated into the input within the span of N text lines. If the macro switches environments (see request "ev") or if the macro contains only control requests, no misplaced underlining will occur.

*III. Mark horizontal place.* A \ka appearing within an input line will result in the then current horizontal position (in the input line) being stored in number register "a".

## X. Hyphenation.

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

<i>Request Form</i>	<i>Initial Value</i>	<i>If no Argument</i>	<i>Cause Break</i>	<i>Explanation</i>
.nh	on	-	no	No hyphenation. Automatic hyphenation is turned off. Words containing hyphens (e. g. mother-in-law) may still be split across lines.
.hy	on	-	no	Hyphenate. Automatic hyphenation is turned on.
.hc c	none	none	no	Hyphenation indicator character is set to "c" or removed. During text processing the indicator is suppressed and will not appear in the output. Prepending the indicator to a word has the effect of preventing hyphenation of that word.

## XI. Three Part Titles.

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

<i>Request Form</i>	<i>Initial Value</i>	<i>If no Argument</i>	<i>Cause Break</i>	<i>Explanation</i>
.tl 'left'center'right'	-	no		Title. The strings represented by "left", "center", and "right" are respectively left adjusted, centered, and right adjusted within the current title length. Any of the fields may be empty. If the character "%" is found within any of the fields it is replaced by the current page number in the current number style (see requests "ar", "ro", and "RO"). Any graphic character may be used in place of the field delimiter "'".
.lt N	65	prev	no	Length of title. The length of lines and titles are maintained separately. Indents do not apply to titles; page-offsets do.

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

## XII. Output Line Numbering.

Automatic sequence numbering of output lines may be turned both on and off. When on, a line number is prepended to output lines. Blank lines and other vertical spaces are not numbered.

- 3 The prepended entity has the general form: (I spaces of number indent) plus (a three digit number with leading zeros printed as spaces) plus (S separating spaces). The prepended entity effectively *offsets* the line which still has the current line length; a reduction in line length is necessary, if the right margin is to be preserved. In addition, it can be specified that only those line numbers that are multiples of some number M are to be printed (the others will appear as blank number fields). The parameters I, S, and M are controllable.

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

- 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 to keep the right side aligned. Some other examples are: ".nm +5 5 x 3" turns on numbering with the line number of the next line to be 5 greater than the last numbered line, with M=5, with spacing S untouched, and with the indent I set to 3; ".np 3" sets M=3 and leaves S and I alone.

### XIII. Conditional Input Line Acceptance

Input lines may be accepted conditionally.

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

#### *Built-in Conditions.*

##### *Name Meaning*

- o The current page number is odd.
- e The current page number is even.
- t The formatter is TROFF.
- n The formatter is NROFF.

Some examples are:

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

outputs a title if the page number is even; and

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

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

### XIV. 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.

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

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



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

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

#### XV. Insertions from the Standard Input Stream

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

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

The contents of "prompt" should be chosen to suggest what or which insertion is currently wanted. Prompting is automatically suppressed when the standard input is not a console typewriter. Ordinarily prompting is used when the output document is being sent to a file as a result of having switched the UNIX Standard Output to a file. A "quiet" mode may be set when NROFF is invoked by giving a "-q" argument that prompts with a bell only and turns off echoing keyboard input so that insertions may be made while the document is being printed. The input text and inserts via "rd" *should not* simultaneously come from the standard input. Multiple copies of a processed "letter" are easily obtained by causing "letter" to reinvoke itself by means of the "nx" request (see below) and including an "ex" request in an insert after the end of the last letter.

#### XVI. Input File Switching

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

<i>Request Form</i>	<i>Initial Value</i>	<i>If no Argument</i>	<i>Cause Break</i>	<i>Explanation</i>
.so filename	-	-	no	Switch Source file. Input at the top (input file) level is switched to the file named "filename". The current input level is not changed. When the new file ends, input is again taken from the original file beginning with the line after the "so" request.

.nx filename	no	Next file is "filename". No further input is taken from the current input file (or current input level); "filename" becomes the current input file and the input level is popped back to the top (input file) level. This request may be used to repeatedly process the same file by having the nextfile be itself.
--------------	----	---

#### XVII. Miscellaneous

<i>Request Form</i>	<i>Initial Value</i>	<i>If no Argument</i>	<i>Cause Break</i>	<i>Explanation</i>
.tm string	-	-	no	Typewriter message. "string" is printed on the user's console typewriter and is <i>not</i> buffered (See § XVIII). "string" is read in copy mode (see § V.II).
.ig	-	-	no	Ignore. Input lines are ignored up to and including the next input line beginning with "...". The ignored lines are read in copy mode and some side effects can occur (e. g. \n+a will increment register "a").
.fl	-	-	yes	Flush output buffer. This request is useful for debugging request sequences because it can be used to force output that typically would be buffered and hidden.
.ab	-	-	no	Abort. This request causes an IOT trap and causes a core image to be produced. It is used for debugging the NROFF program itself.

#### XVIII. Error Messages.

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

## TUTORIAL EXAMPLES

### Introduction

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

### I. Headers and Footers

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

```
.de hd
'sp 6
.ns
..
.de fo
'bp
..
.wh 0 hd
.wh -7 fo
```

describes a header macro named "hd" which produces 1 inch (6 lines) of space (without causing a break), and describes a footer macro named "fo" that will simply eject the page (without causing a break). The "wh" requests specify that "hd" is to be invoked at the beginning of the page (making "hd" a header) and that "fo" is to be invoked after the 7th line from the bottom. The break suppression (using the "'") prevents what is left over from the last line printed on a page from being printed by the occurrence of a break. The ".ns" turns on the no-space mode that suppresses vertical space that would result from the accidental occurrence of an ".sp N" exactly at the location of the intended first output text line on a page.

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

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

```
.nr t 6
.nr b -7
.de hd
'sp \nt
.ns
..
.de fo
'bp
..
.wh 0 hd
.wh \nb fo
```

which achieves the same end except that the margin sizes are initialized in two number registers. Use

ing "\\" in "sp \\\n" results in the stored definition of "hd" containing "sp \\\n" causing the top margin to depend each time on the then current value of number register "t"; this permits easy dynamic modification of the top margin by changing the register "t" (using the "nr" request). Dynamic modification of the bottom margin requires the use of the "ch" request prior to the desired change; for example, ".ch fo \\\nb-3" pushes the trap for "fo" up 3 line spacings. The footer can be arranged to reset itself by including in the definition for "fo" a ".ch fo \\\nb".

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

```
.de fo
'sp 2
.if o .tl "'Bottom Title'
.if e .tl 'Bottom Title'
'bp
..
```

produces differently adjusted titles on even and odd numbered pages. It is often convenient to use something like

```
.de pp
.tl "- % -"
..
.wh -4 pp
```

would place an independently positioned, centered page number a half-inch (after the forth line) from the bottom.

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

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

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

## II. Major Headings and Paragraphs

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

```
.de pg
.sp
.ne 2
.ti 5
..
.de mh
.sp
.ne 4
.ul
..
```

defines a paragraph macro named "pg" which spaces a line, requires that 2 lines of space be left on the page, and sets up a temporary indent of 5 spaces. And defines a major heading macro "mh" that spaces, requires that 4 lines of space be left on the page (room for a one line heading, a space, and two lines of the following paragraph), and arranges for the heading to be underlined.

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

### III. Labeled Indented Paragraphs

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

```
.in N2
.ta N1 N2+1
.ti 0
(tab)label(tab)words in paragraph...
rest of paragraph
.in 0
```

N1 is the distance to the label and N2 is the indent and N2+1 is the corresponding tab stop for filling in the space between the label and the first word of the paragraph. Any spaces within the label would be the unpaddable space (`\(space)`) or characters translated into space using the "tr" request. The Field Mechanism described in §VII.// may be used for more sophisticated paragraph labeling.

### IV. Multiple Column Output

It is relatively easy to arrange for multiple column output. NROFF will generate Reverse-Line-Feed sequences ("`ESC 7`"s) and, if the device used for printing cannot handle such sequences, the COL program may be used as a post-processor\*. The following shows a header and footer set of macros containing the additional requests for generating three column output.

```
.de fo
.if !\\n+c-3 .nc
.if \\nc-3 .np
..
.de nc NEW COLUMN
.po +18+3
.rt
..
.de np NEW PAGE
.po 0
.sp 2
.if !\\n%-1 .tl "- % -"
.bp
..
.de hd
.sp 3
.if !\\n%-1 .tl "- % -"
.sp 2
.ns
.nr c 1 1
.mk
..
.wh 0 hd
.wh -7 fo
.ll 18
(Continued next page.)
```

---

\*See "COL" in the UNIX Programmer's Manual. COL removes reverse-line-feeds by buffering whole pages.

.br

The header macro, "hd", sets a number register, "c", to an initial column number (1) and specifies that any autoincrementing be by 1. The request "mk" marks the place at the bottom of the header as the return point for a subsequent "rt" request. The footer, "fo", increments the column number and tests it to see whether a new column or new page is to be next and invokes either the macro "nc" or "np" accordingly. The new column macro "nc" increments the page offset by 18+3 spaces (line length plus column spacing) and returns to the marked place. The new page macro "np" resets the page offset to the original value and goes on to perform normal footer functions. The line length is set to 18. The three columns are each 18 spaces wide and are spaced 3 spaces apart for a total page width of 66 spaces. The only change necessary to get a different number of columns would be to change the line length, incremental page offset, and the constant against which the column number is compared in the footer macro.

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

## V. Generating Footnotes

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

```
.fn
(Any kind of text and
text processing)
.ef
```

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

A usable footnote program is:

```
.nr bm 7
.de hd HEADER
'sp 3
.tl 'Head title'"
'sp 2
.nr x 0 1
.nr y 0--\\n(bm
.if \\n(dn .fz
.ns
..
.de fo FOOTER
.nr dn 0
.if \\nx .xf
.ch fo --\\n(bm
'bp
..
.de bo
.tl "-- % --"
..
.de fn BEGIN FOOTNOTE
.da FN
.ev1
.if !\\n+x-1 .fs
..
.de ef END FOOTNOTE
.br
(Continued next page.)
```

```
.ev
.di
.nr y -\\n(dn
.ch fo \\ny
.if \\n(nl-\\n(.p-\\ny .ch fo \\n(nl+1
..
.de fs SEPARATOR
-----
.br
..
.de fz
.fn
.nf
.fy
.fi
.ef
..
.de fx
.di fy
..
.de xf
.evl
.nf
.FN
.rm FN
.di
.ev
..
.wh 0 hd
.wh -\\n(bm fo
.wh -4 bo
.ch fo 70
.wh -\\n(bm fx
.ch fo -\\n(bm
.evl
'1155
.ev
.br
```

The size of the bottom margin is specified in number register "bm". The header "hd" initializes two registers, "x" and "y", at the top of every page; "x" is the basic per-page footnote counter and "y" is used during footnote output to keep track of where the footer macro should be sprung. The conditional invocation of macro "fz" reprocesses the remainder of any footnote that did not fit at the bottom of the previous page. The footer tests whether or not any footnotes were processed and if so, invokes "xf". Afterwards the position trap for "fo" is reset to place "bm". The macro "xf" switches environments, sets the "nofill" mode, interpolates in the footnotes (macro "FN"), removes "FN", terminates any possible diversion of footnote material that did not fit on the page, restores fill mode, and restores the previous environment. The macro "fx", also planted at position "bm", will save the portion of the last footnote that didn't fit, if any, by diverting into a macro "fy". The latter is preprocessed in the header, if there has been any diversion (non-zero number register "dn") since the beginning of the footer. The begin-footnote macro "fn" diverts the footnote (in append fashion) into the macro "FN" and switches environments. If the footnote to be processed is the first one on a page, the footnote separator macro "fs" is invoked. In this example, "fs" merely generates a short dashed line. The end-footnote macro "ef" resets the environment, ends the diversion, and pushes up the trap position for "fo" to account for the size of the footnote. If this movement would move the trap up past the next output line (presumably

containing the reference), the trap is moved up only to the latter place; this occurrence ordinarily will result in the last footnote being split between two pages. The "wh" and "ch" requests plant the header trap at the top of the page, plant the footer trap at position "bm", move the footer trap somewhere past the page length, plant a trap for "fx" also at position "bm", and finally move the footer trap back. The two macros "fo" and "fx" are effectively planted at the same place; the trap for "fx" can occur only if the footer trap is moved up by the occurrence of a footnote, because it is further down the internal trap list; it was necessary to temporarily move the trap for "fo" to avoid "fx" replacing "fo" at that trap position.



5/19/75

## Quick NROFF Addendum

### Arguments

- raN     Number register "a" is set to N.
- mx     Simulates ".so /usr/lib/tmac.x"; "x" can be multicharacter.

### Escapes

- \ "     This escape and the rest of the input line are ignored. May be used anywhere. If used at beginning of a line the line acts like a blank line.

### Read only registers

- .c     input line count in current file.
- .h     high-water mark of register "nl" on current page.
- .n     length of text on last output line.

### Functions

- \w'string'  
     Width of "string".

### Other

Sentences ending with "!" and "?" are given an extra terminal space just as with ".".

### New or changed requests

- .eo     Escape processing is turned off. May be turned on again using ".ec".
- .rn xx yy     Rename request/macro "xx" to "yy".
- .pi prog     Pipe the output to the program "prog". No arguments transmitted. Must occur before NROFF unloads it's first buffer.
- .hy N     Control hyphenation where N is:  
         0 = don't hyphenate.  
         1 or greater = hyphenate.  
         2 = don't hyphenate last lines (more specifically lines that will cause a trap).  
         4 = don't do -xx| (split off last two characters).

5/19/75

8 = don't do |xx- (split off first two).  
These are additive; i.e. .hy 12 invokes last two restrictions.

.hw word1 word2 ...

Specify hyphenation exception list (words contain "-" indicating where hyphens should go). 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. Space available is small - about 128 characters total.

.af xx c

assigns a format to the number register xx. "c" may be

1 for 0,1,2,3,4,...

i for 0,1,ii,iii,iv,...

I for 0,I,II,III,IV,...

a for 0,a,b,c,...,z,aa,ab,...,zz,aaa,...

A for 0,A,B,C,...,Z,AA,AB,...,ZZ,AAA,...

If "c" is missing or not recognized, the format assignment is removed. Number registers without a format assignment are subject to "ar", "ro", and "RO". The read-only registers do not have a format. The number returned by \w'...' is always arabic.

.cu N

Continuously underline the next N lines.

.mc c N

Specify a margin character to appear after each non-empty text line (except those produced by ".tl"); the margin character "c" is printed in the right margin separated by N spaces (default 2) from the line length as exemplified by this paragraph. If the output line is too-long (as can happen in nofill mode) the character will appear with N=0. If there are no arguments, the mechanism is turned off.