UNED
1004

COVER SHEET FOR TECHNICAL MEMORANDUM

TITLE- NROFF User's Manual

MM-73-1271-2

CASE CHARGED- 39199

DATE- February 5, 1973

FILING CASES- 39199-11

AUTHOR- J. F. Ossanna
MH-2C-574
MH-3520

FILING SUBJECTS- Text Formatting

ABSTRACT

NROFF is a text formatter on Center 127's PDP-11/45 UNIX Time-Sharing System. This User's Manual consists of: a description of usage; a Request Summary and Index; a Reference Manual keyed to the index; and a set of Tutorial Examples. NROFF accepts text interspersed with lines of format control information and produces a printable, paginated document having a user-designed style. Examples of styling freedom include: arbitrary style headers and footers; arbitrary style footnotes; multiple automatic sequence numbering; and multiple column output (with the aid of a post processor). Basic mechanisms available include: macros; macro expansion on line traps; number registers; multiple text processing environments; conditional input; and form letter ability.

Text - 28 pages
Figure 1

UNED (E)

SEE REVERSE SIDE FOR DISTRIBUTION LIST

MM-73-1271-2
February 5, 1973

J. F. Ossanna

NROFF USER'S MANUAL

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**).

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. The options which may appear in any order so long as they appear before the files are:

Option	Effect
+N	Output will commence at the first page whose page number is N (independent of whether or not the page number is being printed).
-M	Output will end after the first page whose page number is M.
-s	Stop between pages. Printing will halt prior to each page (including the first) to permit paper loading and changing. Printing is restarted by typing either a "newline" or "delete" character.
-h	High-speed output. During output, strings of space characters are replaced where possible with tab characters to speed up output. If the output

---

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

**See description of "ov" in the UNIX User's Manual and tutorial part of this manual.

is directed into a file or a pipe***, this mode reduces the total number of characters in the file or pipe.

-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).

-i Index mode. NROFF creates a file called "index" containing every word output together with the line and page number. The format is word, tab, page, tab, line, newline, etc. Invoking this mode slows down the execution of NROFF considerably.

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.

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

***See description of "pipes" in the UNIX User's Manual.

REQUEST REFERENCE AND INDEX

Request Form	Initial Value	If no Argument	Cause* Break	Explanation
I. Page Control				
.pl ±N	N=66	N=66	no	Page Length.
.bp ±N	N=1	-	yes	Begin Page.
.pn ±N	N=1	ignored	no	Page Number.
.po ±N	N=0	N=prev	no	Page Offset.
.ne N	-	N=1	no	NEed N lines.
II. Text Filling, Adjusting, and Centering				
.br	-	-	yes	BReak.
.fi	fill	-	yes	FIll output lines.
.nf	fill	-	yes	NoFill.
.ad c	adj,norm	adjust	no	ADjust mode on.
.na	adjust	-	no	NoAdjust.
.ce N	off	N=1	yes	CEnter N input text lines.
III. Line Spacing and Blank Lines				
.ls ±N	N=1	N=prev	no	Line Spacing.
.sp N	-	N=1	yes	SPace N lines
.lv N	-	N=1	no	OR-
.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.
IV. Line Length and Indenting				
.ll ±N	N=65	N=prev	no	Line Length.
.in ±N	N=0	N=prev	yes	INdent.
.ti ±N	-	N=1	yes	Temporary Indent.
V. Macros, Diversion, and Line Traps				
.de xx	-	ignored	no	DEfine or redefine a macro.
.rm xx	-	-	no	ReMove macro name.
.di xx	-	end	no	DIvert output to macro "xx".
.wh ±N xx		-	no	WHen; set a line trap.
.ch ±N ±M		-	no	OR-
.ch xx ±M		-	no	OR-
.ch ±N y		-	no	OR-
.ch xx y		-	no	CHange trap line.

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

Request Form	Initial Value	If no Argument	Cause Break	Explanation
VI. Number Registers				
.nr a ±N =M		-	no	OR-
.nr ab ±N =M		-	no	Number Register.
.nc c	\n	\n	no	Number Character.
.ar	arabic	-	no	Arabic numbers.
.ro	arabic	-	no	Roman numbers.
.RO	arabic	-	no	ROMAN numbers.
VII. Input and Output Conventions and Character Translations				
.ta N,M,...		none	no	PseudoTAbs setting.
.tc c	space	space	no	Tab replacement Character.
.lc c	.	.	no	Leader replacement Character.
.ul N	-	N=1	no	UNderline input text lines.
.cc c	;	;	no	Basic Control Character.
.c2 c			no	Nobreak control character.
.li N	-	N=1	no	Accept input lines LIterally.
.tr abcd....		-	no	TRanslate on output.
VIII. Hyphenation.				
.nh	on	-	no	No Hyphen.
.hy	on	-	no	HYphenate.
.hc c	none	none	no	Hyphenation indicator Character.
IX. Three Part Titles.				
.tl 'left'center'right'			no	TitLe.
.lt N	N=65	N=prev	no	Length of Title.
X. Output Line Numbering.				
.nm ±N M S I		off	no	Number Mode on or off, set parameters.
.np M S I		reset	no	Number Parameters set or reset.
XI. Conditional Input Line Acceptance				
.if c anything		-	no	OR-
.if !c anything		-	no	OR-
.if N anything		-	no	OR-
.if !N anything		-	no	IF true accept line of "anything".
XII. Environment Switching.				
.ev N	N=0	N=prev	no	EnVironment switched.
XIII. Insertions from the Standard Input Stream				
.rd prompt		bell	no	ReaD insert.
.ex	-	-	no	EXit.

Request Form	Initial Value	If no Argument	Cause Break	Explanation
XIV. Input File Switching				
.so filename	-		no	Switch SOurce file (push down).
.nx filename	-		no	NeXt file.
XV. Miscellaneous				
.ig	-	-	no	IGnore.
.fl	-	-	no	FLush output buffer.
.ab	-	-	no	ABort.

Alphabetical Request List with Section Numbers

ab	XV	in	IV	pn	I
ad	II	lc	VII	po	I
ar	VI	li	VII	rd	XIII
bp	I	ll	IV	rm	V
br	II	ls	III	RO	VI
c2	VII	lt	IX	ro	VI
cc	VII	lv	III	rs	III
ce	II	na	II	so	XIV
ch	V	nc	VI	sp	III
de	V	ne	I	sv	III
di	V	nf	II	ta	VII
ev	XII	nh	VIII	tc	VII
ex	XIII	nm	X	ti	IV
fi	II	np	X	tl	IX
fl	XV	nr	VI	tr	VII
hc	VIII	ns	III	ul	VII
hy	VIII	nx	XIV	wh	V
if	XI	os	III	xh	III
ig	XV	pl	I		

Explanation

Input lines beginning with a "." are interpreted as format control lines containing a format request. Undefined requests are completely ignored. The "break" function associated with any request can be suppressed by using the no-break control character "'" instead of "." to indicate control lines. A "break" is the forced output of a partially filled line.

In the following discussion numerical arguments are presented in several symbolic forms: ±N means that the argument may take the forms N, +N, or -N and that the corresponding effect is to set the affected parameter to N, to increment it by N, or decrement it by N respectively; ±N means that the argument may take the form N or -N and that the effect is to set the affected parameter to N or -N. Other capital letters are used for additional numerical 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.

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.

I. Page control

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

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.pl ±N	N=66	N=66	no	Page Length set to N lines (total lines on whole page).
.bp ±N	N=1	-	yes	Begin Page. The current page is ejected and a new page is begun. If ±N is given, the new page number will be ±N. See request ns.
.pn ±N	N=1	ignored	no	Page Number. The next page (when it occurs) will have the page number ±N.
.po ±N	N=0	N=prev	no	Page Offset. ±N spaces are prepended to each output line; i. e. the page image is moved ±N spaces to the right.

.ne N	-	N=1	no	NEed N lines. If the distance, D, to the next trap line is less than N, D blank lines are output. If there are no remaining traps on the page, D will be the distance to the bottom of the page. See Section V.

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. 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 blanks 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.

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.br	-	-	yes	Causes a BReak. The filling of the line currently being filled out is stopped and the line is printed. Text lines beginning with spaces or tabs converting to spaces also cause a break. An empty text line (blank line) causes 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"). Normal input and output translations occur (see Sections VI and VII).
.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.

III. Line Spacing and Blank Lines

The default line spacing is single space. Line spacing may be dynamically set and reset to provide any desired spacing. When not single spacing, the additional blank lines are put out after each output text line. Blocks of one or more blank lines can be requested.

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.ls ±N	N=1	N=prev	no	Line Spacing is set to ±N. If the line spacing is N, N-1 blank lines are put out after each generated output line. The occurrence of a line trap will terminate spacing.
.sp N	-	N=1	yes	SPace N lines or to the next trap line or to the bottom of the page, whichever is less. If the no-space mode is on, no spacing occurs.
.lv N	-	N=1	no	OR-
.sv N	-	N=1	no	SaVe N lines. If the distance to the next trap line is equal to or more than N lines, N lines are output. No-space mode has no effect. If this distance is less than N lines, no lines are immediately output, but N is remembered for later output (see request "os").
.os	-	-	no	Output Saved block of 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	-	-	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 (see Section II).

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.ll ±N	N=65	N=prev	no	Line Length is set to ±N.
.in ±N	N=0	N=prev	yes	INdent is set to ±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 L-N print positions.
.ti ±N	-	N=1	yes	Temporary Indent. The next output text line will be indented ±N spaces (±N with respect to the current indent). The current indent is not changed.

V. Macros, Diversion, and Line Traps

A macro is a named set of one or more lines that may be invoked by name or by the advent of having output the Nth line on a 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"). A macro named "ab" may be invoked by an input line beginning with ".ab"; the rest of the line may contain up to nine argument strings. In addition, a trap may be set on the Nth output line on each page to invoke the macro (see "wh"). Macros may contain arbitrary request and text lines.

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.de xx	-	ignored	no	DEfine or redefine a macro with the one or two character name "xx". The contents of the macro begins on the next input line. Input lines are copied until the definition is terminated by a line beginning with "..". No interpretation of the input occurs except that normal input translations occur (see Section VII). A macro may contain "de" requests provided the corresponding ".." is concealed to prevent copy termination; "\\.." will copy as "\.." and be reread as "..".
.rm xx	-	-	no	ReMove macro name. The macro name "xx" is removed from the request name list. Subsequent invocations of the macro will have no effect unless the name is defined again.
.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" is encountered without an argument. Diversions cannot be nested. The number of output lines created during diversion is kept in a number register (named "dn") for possible later use (see request "ch").
.wh ±N xx	-		no	WHen (after) output line ±N occurs invoke macro "xx". Any previous

macro planted at line ±N is replaced by "xx". N is the line number measured from the top beginning with one. -N refers to the Nth line from the bottom; if the page length is 66, line -1 is line 66. N=0 will set a trap at the beginning of a page and must be used for headers. If no macro name is given, the trap planted at line ±N, if any, is removed.

.ch ±N ±M	-	no	OR-
.ch xx ±M	-	no	OR-
.ch ±N y	-	no	OR-
.ch xx y	-	no	CHange the trap planted at line ±N to occur instead at line ±M. Alternatively, change the line at which the macro "xx" is planted to be ±M. If no trap exists at line ±N, the request is ignored. If instead of ±M, some non-numeric character "y" is given, the trap line number is decremented by the number of lines resulting from the most recent output diversion, unless such decrementing would result in a number corresponding to a line on the current page prior to or equal to the current number of lines output plus the current line spacing.

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 Section VII). If an argument contains blanks, it must be surrounded by double-quotes. For example,

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

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

```
\$N
```

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

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

and called by

```
.xx Monday 14 th
```

to produce the text

Today is Monday the 14th.

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. A number register is invoked by the sequence

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

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

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.nr a ±N =M	-		no	OR-
.nr ab ±N =M	-		no	Number Register definition or modification. a or ab is the register name. The register is (re)set to ±N. The increment to be used for auto-incrementing is set to =M.
.nc c	\n	\n	no	Number Character is set to "c". For example, if c = X, Xa will invoke the register a.

.ar	arabic	-	no	OR-
.ro	arabic	-	no	OR-
.RO	arabic	-	no	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: "%" is the current page number; "dn" is the number of output lines processed during the last diversion (see request "di"); and "nl" is the number of lines actually output on the current page.

VII. Input and Output Conventions and Character Translations

Certain character translations occur both when the input file is read and when stored macros are invoked and reread.

Tabs and leaders. Horizontal tab characters are replaced by strings of space characters according to the current table of pseudotab settings. In addition, a leader character (ascii "soh") is replaced by strings of dots (".") according to the same pseudotabs. A tab or leader character occurring on or after the largest pseudotab stop is replaced by one space or dot respectively. The characters used in the string replacments and the pseudotab settings may be changed.

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.ta N,M,...		none	no	PseudoTAbs are set to columns N,M,... Default pseudotabs are set to 9,17,25,...,73 (i. e. tabs are worth 8 spaces); a maximum of 13 pseudotab stops may be set. Tabs occurring at columns beyond the current tab table are replaced by a single character. The value N,M,etc may be separated by commas, spaces, or any other nonnumerics.
.tc c	space	space	no	Tab replacement Character is set to "c" or reset to space.
.lc c	.	.	no	Leader replacement Character is set to "c" or reset to ".".

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.

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.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.

Control characters (".", ","). Ordinarily input lines beginning with . are taken to be NROFF request (control) lines. Lines beginning with "," are also taken to be request lines except that the request is inhibited from causing a break. Both of these 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.

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

Another way to have lines beginning with control characters interpreted as text is to use the "tr" request (described later).

Escape conventions. A number of graphical escape conventions are provided to enable inputting characters ordinarily transformed, graphical input of nongraphical characters, and inputting some non-typeable characters. In the following (esc) represents the "escape" control character, (so) is "shift out", (si) is "shift in", and (nl) is "newline".

Actual Input	Interpretation	Explanation
\d	(esc)9	Half-line forward.
\u	(esc)8	Half-line reverse.
\r	(esc)7	Full-line reverse.
\x	(so)	Shift out (extra characters).
\y	(si)	Shift in (normal characters).
\t	Tab	Untransformed horizontal tab.
\l	Delete	Delete character.
\n	....	Number register follows (unless redefined; see request "nc").
\$	....	Argument indicator (see Section V).
\!	....	Transparent throughput indicator (see later).
\\	\	Backslash (uninterpreted as graphical escape).
\(nl)	ignored	Concealed newline character.

A \ followed by any character other than those above is ignored; for example, "\a" is just "a".

Model 37 teletypewriter functions represented by (esc)-character sequences are retained provided the character is 1, 2, 3, 4, 7, 8, or 9.

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*\na+2/13 evaluates to -2. It should be remembered that in certain contexts an initial + or - is taken as an incremental designator.

Output translation. Provision exists for specifying a mapping of any character into any other character. All text processing takes place with the original character and the translation occurs at the moment of output.

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.tr abcd....	-		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 space.

A common use of the "tr" request is to provide nonadjustable spaces. When normal filling and adjusting is done the space between words is what is adjusted to adjust a line and is what indicates 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". If the tab replacement character is changed to one that is in turn output translated into space, then tabs effectively produce nonadjustable space.

Transparent Throughput. An input line beginning with a "\!" is transparently output (except for the initial "\!"); no translations occur, nor does any other processing take place. This mechanism is used to pass control information to a post-processor or to pass NROFF requests to a macro during a diversion.

VIII. Hyphenation.

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

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.nh	on	-	no	No Hyphen. 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.

IX. Three Part Titles.

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

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.tl 'left'center'right'			no	TitLe. The strings represented by "left", "center", and "right" are respectively left adjusted, centered, and right adjusted within the current title length. Any of the fields may be empty. If the

character "%" is found within any of the fields it is replaced by the current page number in the current number style (see requests "ar", "ro", and "RO"). Any graphic character may be used in place of the field delimiter "'".

.lt N	N=65	N=prev	no	Length of Title. The length of lines and titles are maintained separately.

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

X. Output Line Numbering.

Automatic sequence numbering of output lines may be turned both on and off. When on, a line number is prepended to the otherwise unaffected output lines. Blank lines are not numbered. The prepended entity has the general form: (I spaces of number indent), (a three digit number with leading zeros printed as blanks), (S separating spaces). 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.

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

As an example, the paragraph portions of this section are numbered with M=3: ".nm 1 3" was placed at the beginning; ".nm" was

placed at the end of the first paragraph; and ".nm +0" was placed in 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.

XI. Conditional Input Line Acceptance

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.if c anything	-	no	OR-	
.if !c anything	-	no	OR-	
.if N anything	-	no	OR-	
.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.

Some examples are:

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

outputs a title if the page number is even; and

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

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

XII. Environment Switching.

A number of the parameters that control the text processing are gathered together into an "environment" which can be switched by the user.

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.ev N	N=0	N=prev	no	EnVironment is switched to environment N; 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:

line spacing	underline count	request control chars
line length	centering count	number register indicator char
indenting	line numbering	tab replacement char
adjusting	pseudo tab table	leader replacement char
filling	hyphenation control	partially collected lines
title length		

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 lines are kept with an environment so that environment switching prevents the next break from printing the previous environment's partial line.

XIII. Insertions from the Standard Input Stream

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

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.rd prompt		no prompt	no	ReaD insert. NROFF input is switched to UNIX Standard Input until two newline characters in row are found. The extra newline is thrown away. A "delete" character will also end insertion. If "prompt", an arbitrary character string without blanks, is given, "prompt" is written out on the

				user's typewriter to indicate that the input is requested. Because "rd" behaves like a macro, arguments may be placed after "prompt".
.ex	-	-	no	EXit. Text processing is finished exactly as if the input file ended and there were no more input files.

The contents of "prompt" should be chosen to suggest what or which insertion is currently wanted. 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. Prompting is automatically suppressed if the UNIX Standard Input has been switched to a file.

If the text to be formatted is in the file "letter" and has "rd" requests,

```
nroff letter >doc
```

invokes normal prompting and puts the resulting document into the file "doc".

```
nroff -q letter
```

prompts with a bell and allows inserting while the document is being printed.

```
nroff letter <inserts
```

suppresses prompting and reads the inserts from the file "inserts". 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.

XIV. Input File Switching

At any given instant, input is taken from either the current input file (the top input level) or from a macro (at some macro invocation level).

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.so filename	-		no	Switch SOurce file. Input at the top (input file) level is switched to the file named "filename". The current input level is not changed. When the new file ends, input is

again taken from the original file beginning with the line after the "so" request.

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

XV. Miscellaneous

Request Form	Initial Value	If no Argument	Cause Break	Explanation
.ig	-	-	no	IGnore. Input lines are ignored up to and including the next input line beginning with "..".
.fl	-	-	no	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 NROFF debugging.

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 6 blank lines (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 after line 0 (making "hd" a header) and that "fo" is to be invoked after the 7th line from the bottom (producing 6 blank lines). 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 blank lines that would result from the accidental occurrence of an ".sp N" exactly at the location of the intended first output text line on each page or from a line spacing greater than one.

The line trap at line zero on 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 th 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. Using "\\" in "'sp \\nt" results in the stored definition of "hd" containing "'sp \nt" causing the top margin to depend each time on the then current value of number register "t"; this permits easy dynamic modification of the top margin by changing the register "t" (using the "nr" request). Dynamic modification of the bottom margin requires the use of the "ch" request prior to the desired change; for example, ".ch fo \nb-3" pushes the line trap for "fo" up 3 lines. The footer can be arranged to reset itself by including in the definition for "fo" a ".ch fo \\nb".

With the above kind of footer something like

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

would place an independently positioned, centered page number on th 3rd line from the bottom.

A more typical header macro might look like

```
.de hd
'sp 3
.tl ''- % -''
'sp 2
.ns
..
```

and produce a centered page number at the top of the page. Or like

```
.de hd
'sp 2
.if e .tl 'Page %'''
.if o .tl '''Page %'
.tl ''Some Title''
'sp 2
.ns
..
```

which produces a left adjusted page number on even numbered pages, a right adjusted number on odd pages, and centers a title on the next line.

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

```
.de fo
'sp 2
.tl ''Bottom Title''
'bp
..
```

is a simple example.

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

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

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

II. Multiple-Column Output

Multi-column output is achieved by having NROFF produce output that can be folded or overlayed to obtain page images with more than one column. An overlay program, OV, may be invoked at UNIX command level or as a pipe. OV reads an input file (or pipe) assumed to contain pairs of 66 line page components and literally overlays successive page component pairs.

The attached Figure 1 shows an example of page layout that can be used to prepare output for overlaying with OV to obtain double column output. If the output page length is to be P lines, the page length used during NROFFing would be 2P lines. The top half of this double length pseudopage is generated with the normal header, with a column-width line length, and with a zero indent. A line trap is set at the last line of column one to invoke a number of blank lines

that will later overlay the real header and footer, and to set the indent to offset the second column. The bottom half of the pseudo-page is generated with an indent equal to the first column width plus the desired spacing between columns and a line length equal to two column widths plus the spacing. The header must reset the indent. Any indent control within the text must be incremental in form to avoid affecting the column indenting. The following set of requests would arrange a page layout corresponding to that of Figure 1 (with an output page length of 66 lines):

```
'pl 66*2
.de hd
'sp 3
.if \\n%-1 .tl ''- % -''
.if !\\n%-1 'sp
'sp 2
'll -35
'in -35
.ns
..
.de fo
'bp
..
.de pp
.tl ''- % -''
.rm pp
..
.de im
'sp 6+6
'll +35
'in +35
.ns
..
.wh 0 hd
.wh -7 fo
.wh -66-7 im
.wh -4 pp
.br
```

Header "hd" is six lines long, produces a centered page number on every page but the first, subtracts 35 (column width plus column spacing) from the indent to change back to column one (the indent cannot be set negative so this effectively sets it to zero the first time) and similarly decrements the line length. The macro "im" produces the 12 blank lines that will overlay the real header and footer, and increments the indent by 35 spaces. Footer "fo" merely ejects the page. Macro "pp" places a centered page number at the bottom of the first page and removes itself. The actual overlaying can be done two ways. The command sequence

```
nroff files >temp
ov temp
rm temp
```

directs the output from NROFF into a temporary file "temp" which is then used as input for OV; afterward "temp" is removed. The same end can be achieved more easily by

```
nroff files >ov>
```

which uses the UNIX pipe mechanism to direct the output of NROFF to the input of OV.

Because of the overlaying, many page layout alternatives can achieve the same end. For example, manipulation of the page offset (see "po" request) or the number indent (see "nm" request) can be used to separate columns. In addition the basic approach can be extended to any number of columns. Four column output, for example, is obtained by overlaying twice, and eight columns by overlaying three times. The following request set arranges for three column output; the details are left for the reader to examine.

```
'pl 66*4
.nr i 18
.nr s 5
.de hd
'sp 3
.tl ''= % -''
'sp 2
'll -\\ni+\\ns*2
'in -\\ni+\\ns*2
.ns
..
.de fo
'bp
..
.de im
'sp 6+6
'll +\\ni+\\ns
'in +\\ni+\\ns
..
.wh 0 hd
.wh -66*3-7 im
.wh -66*2-7 im
.wh -66-7 fo
.br
```

The total processing is accomplished by

```
nroff files >ov>ov>
```

III. 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 fol-

lowing

```
.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, anf "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 m -7
.de hd
'sp3
.tl ''- % -''
'sp2
.nr x 0 1
.nr y 0 1
.if \n(dn .if \nz .fz
.nr z 0
.ns
..
.de fz
.fn
.nf
.fy
.fi
.ef
..
.de fo
.ev2
.nf
.if \nx .xf
.di
.ev
.ch fo \nm
'bp
..
.de xf
.\n+y
.rm \ny
.if \nx-\ny .xf
..
(cont. next col.)
```

```
.de fx
.di fy
.nr z 1
..
.de fn
.di \n+x
.ev1
.if !\nx-1 .fs
..
.de fs
.ti0
----------
.br
..
.de ef
.br
.ev
.di
.ch fo x
..
.wh 0 hd
.wh -7 fo
.ch fo 500
.wh -7 fx
.ch fo -7
```

The size of the bottom margin is specified in number register "m". 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. 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 switches environments

and then tests whether or not any footnotes were processed and if so, invokes "xf". Afterwards the line trap for "fo" is reset to line "m". The macro "xf" expands each footnote, removes it, and, if any remain, it invokes itself. The macro "fx", planted at line "m", will save the portion of the last footnote that didn't fit. The begin-footnote macro "fn" diverts the footnote to a macro whose name is generated by autoincrementing register "x", and switches environments. If the footnote to be processed is the first one on a page, the footnote separator macro "fs" is invoked. The end-footnote macro "ef" resets the environment, ends the diversion, and pushes up the line trap for "fo" to account for the size of the footnote. The "wh" and "ch" requests plant the header trap at line zero, plant the footer trap at line "m", move the footer trap somewhere past the page length, plant a trap for "fx" at line "m", and finally move the footer trap back. The two macros "fo" and "fx" are effectively planted at the same line; 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 line trap list; it was necessary to temporarily move the trap for "fo" to avoid "fx" replacing "fo" at that trap.

It is left for the reader to examine the further details of this example.

Joseph F. Ossanna

J. F. OSSANNA

MH-1271-JFO-gam

Attached
Figure 1

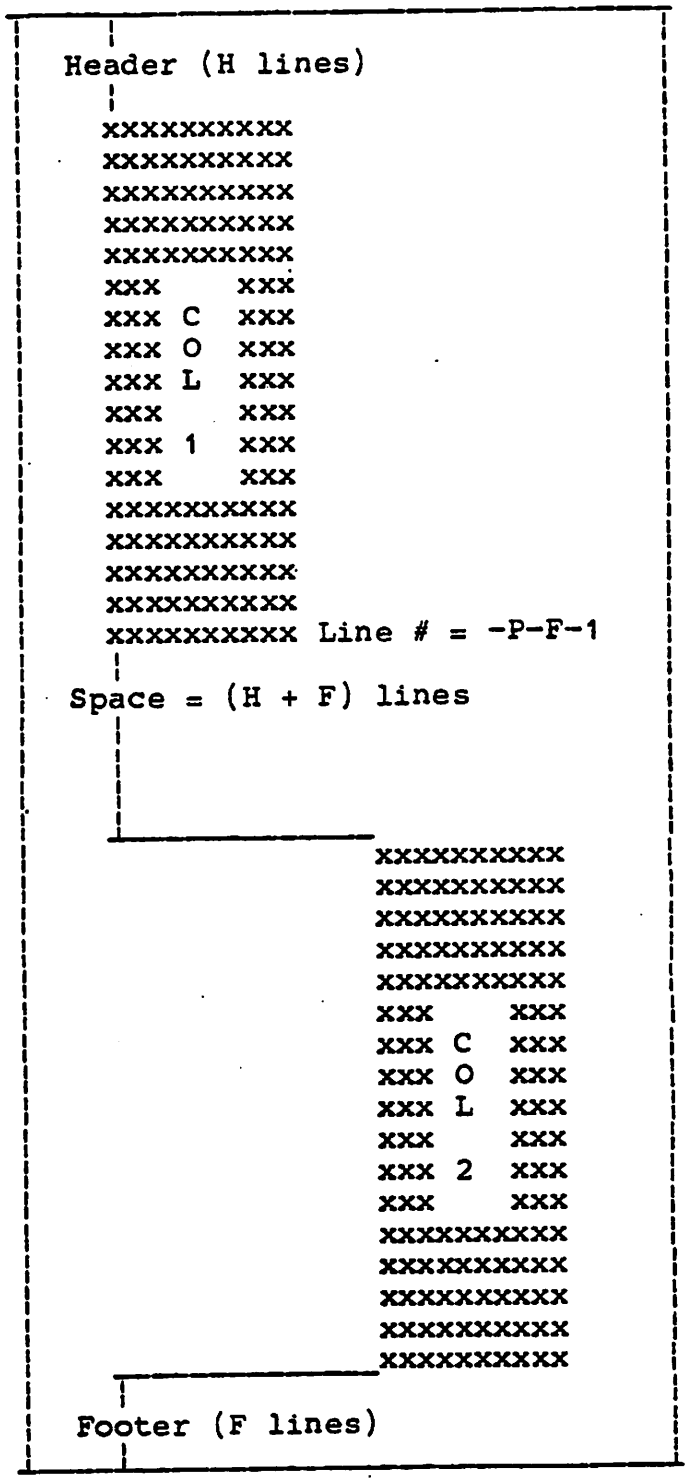

Fig. 1. NROFF page layout (length 2P lines) for overlay production of double column output (page length P lines).