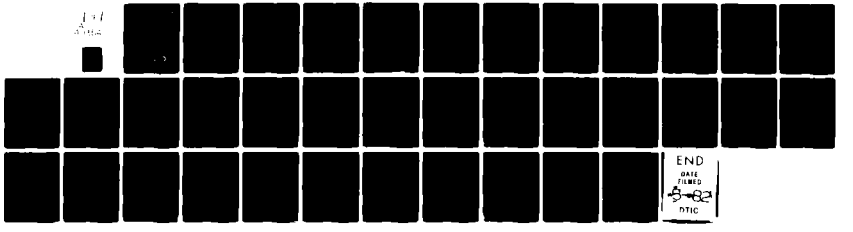
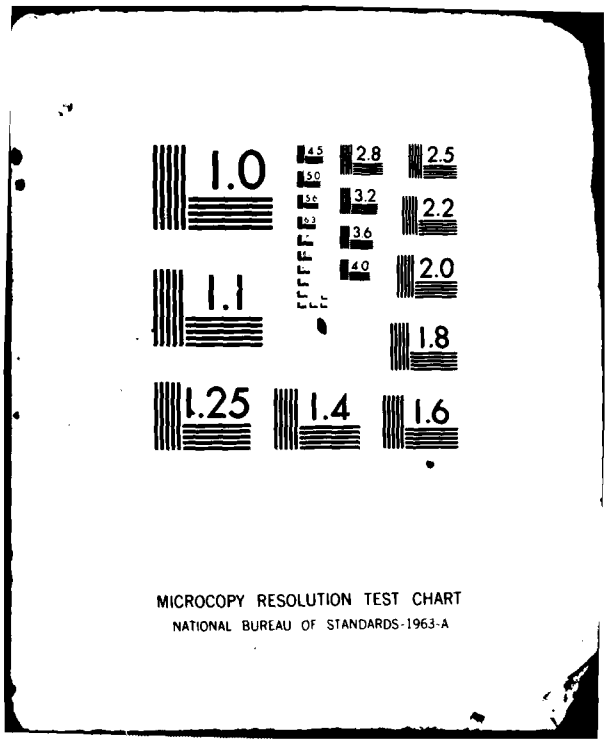
AD-A113 541 DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CE--ETC F/G 5/2
AN INTRODUCTION TO THE '-ME' MACROS USED WITH NROFF ON THE VAX --ETC(U)
MAR 82 D J HIBBERT
UNCLASSIFIED DTNSRDC/CMLD-82-07 NL



END
DATE FILMED
5-82
DTIC


1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(12)

CMLD-82-07

DAVID W. TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER

Bethesda, Maryland 20084

AD A113541

DTIC FILE COPY

An Introduction to the "-ME" Macros Used with NROFF on the VAX 11/780

AN INTRODUCTION TO THE "-ME" MACROS USED WITH NROFF ON THE VAX 11/780

by

Dabney J. Hibbert

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

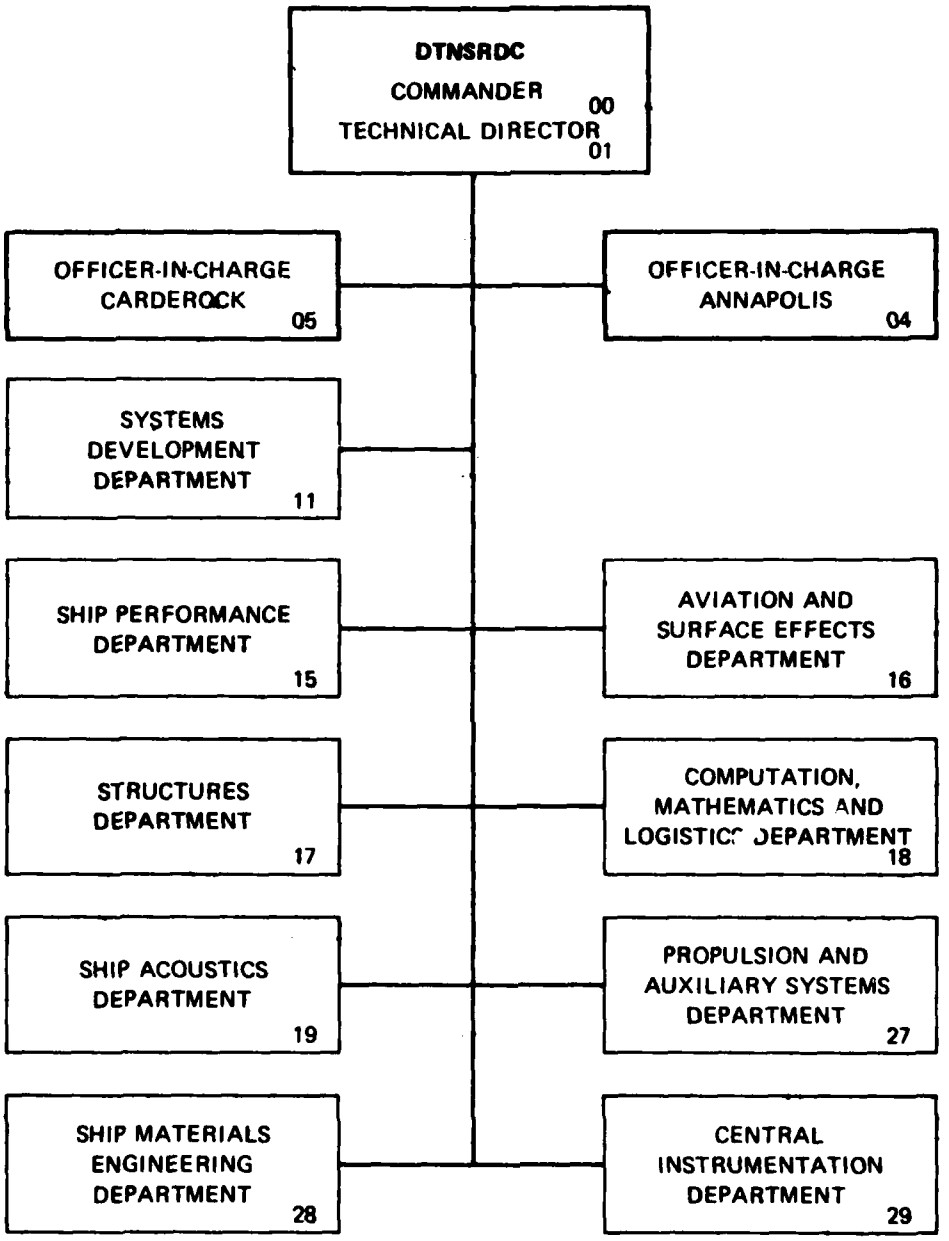
Computation, Mathematics and Logistics Department
Departmental Report

March 1982

CMLD-82-07

NDW-DTNSRDC 5602/30 (2-80)
(supersedes 3960/46)

MAJOR DTNSRDC ORGANIZATIONAL COMPONENTS



GPO 866 987

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: CMLD-82-07	2. GOVT ACCESSION NO.: AD-A113 541
	3. RECIPIENT'S CATALOG NUMBER:
4. TITLE *(and Subtitle)*: An Introduction to the "-ME" macros used with NROFF on the VAX 11/780	5. TYPE OF REPORT & PERIOD COVERED: Final
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Dabney J. Hibbert	8. CONTRACT OR GRANT NUMBER(s):
9. PERFORMING ORGANIZATION NAME AND ADDRESS: David W. Taylor Naval Ship R&D Center, ADP Software Specialist (Code 189.1), Bethesda, Maryland 20084	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS:	12. REPORT DATE: March 1982
	13. NUMBER OF PAGES: 33
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*:	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for Public Release: Distribution Unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Text processing, memorandum macro instructions, macro instructions, UNIX, NROFF, input file, headers, footers, lists, displays, delayed text.

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This document gives document preparation users a brief introduction to and provides some examples of "-ME" macros, a general purpose package of text formatting macros used with the UNIX text formatter, "NROFF". These macros provide a flexible tool for producing many common types of documents.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This document gives document preparation users a brief introduction to and provides some examples of "-me" macros, a general purpose package of text formatting macros used with the UNIX text formatter, NROFF. These macros provide a flexible tool for producing many common types of documents.

PURPOSE

This document describes the text processing facilities available on the UNIX[*] operating system on the DEC VAX 11/780 via the NROFF[*] text formatter and its -me macro package[**]. This document will also attempt to show the major differences between the -me macro package and the -mm macro package, which was used on the DEC 11/70.

NROFF, a computer program that runs under the UNIX operating system, reads an input file prepared by the user and outputs a formatted document. The input file consists of both text and macro instructions, which indicate to the NROFF program how the user wishes the output to be formatted.

DIFFERENCES BETWEEN NROFF MACRO PACKAGES ON THE VAX AND ON THE 11/70

There are differences between most of the NROFF memorandum macro instructions used on the DEC 11/70 as compared to those used on the VAX 11/780. For those users who have used NROFF and its associated memorandum macros (i.e., the "-mm" option) on the DEC 11/70, Figure 1 should prove to be a useful quick reference guide. This figure presents a comparison chart detailing the more commonly used NROFF instruction macros as used on both the 11/70 and the VAX 11/780.

[*]UNIX and NROFF are Trademarks of Bell Laboratories
[**]Developed at the University of California, Berkeley

	11/70	VAX
spacing	.sp n	.sp n
centering	.ce n	.ce n
paragraphing	.P	.pp
break page	.bp	.bp
headers (example is for blanks at top)	.PH "''''"	.he ''''
footers (example is for -n- at bottom)	.PF "''-\\\\nP''"	.fo ''-%-''
underlining		.ul n
lists	.AL a(or 1,A,etc,) with .LI,.LI,....LE	.ip .np
double spaces		.ls 2
single spaces (default unless changed)		.ls 1
line indent	.in +n	.in +n
2 Column	.2C	.2c
1 Column (default unless changed)	.1C	.1c
Start 2nd col.		.bc
Unnumbered Sections	.HU "_____"	.uh "_____"
Numbered Sections	.H1 "_____" .H2 "_____", etc.	.sh 1 "_____" .sh 2 "_____", etc.

Figure 1 Comparison of Functions of NROFF Macro Packages on the 11/70 and the VAX

MANUALS USEFUL FOR DOCUMENT PROCESSING

(1) NROFF/TROFF User's Manual - This is an assembly language level manual and should be used by those who want to change existing macros or write new ones.

(2) Writing Papers with NROFF Using -ME - Good reference manual that contains examples.

(3) -ME Reference Manual - Quick reference manual for the memorandum macros.

GUIDELINES FOR TEXT PROCESSING

Because NROFF is a text formatter, it is responsible for formatting the text. That is, NROFF collects words from the input lines and fills the output lines, justifying the right hand margin by inserting extra spaces in the line. In addition, NROFF will perform more complex tasks for the user, such as automatically numbering pages and skipping over page folds on a printer device. NROFF will also handle footnote placement, index preparation, and reference quotations. The following lists a few basic guidelines for text processing.

1. All instructions to NROFF (macro instructions) are placed on a separate line within the text and are preceded by a "." (period).

2. It is recommended that those preparing input text follow the suggestions below:

   (a) Begin each new sentence on a separate line, since common corrections are to add or delete sentences.

   (b) Do not put spaces at either the beginning or the end of lines.

   (c) Do not hyphenate words at the end of lines.

   (d) Do not break words that should have hyphens in them (such as mother-in-law) over a line - instead, start a new line before the word.

3. An example of the above rules is given below:

```
.pp
The project team approach allows for better
dissemination of technical and status information.
This approach also allows for a certain amount of
on-the-job training when a team consists of both
trainee and journeyman personnel.
.sp2
```

The above input lines will be formatted by NROFF and will produce the following output:

The project team approach allows for better dissemination of technical and status information. This approach also allows for a certain amount of on-the-job training when

a team consists of both trainee and journeyman personnel.

4. After the input file (consisting of document text and macro instructions) is ready to be output, the user calls the NROFF formatter by the command:

nroff -me yourfile

This command will display the formatted text on your CRT screen. To have the document printed on a printer device, (such as a Diablo 1650 or a LA120), the following command is used:

nroff -me yourfile|cat -d

PARAGRAPHS

Paragraphs are begun by using the ".pp" instruction. For example, the input:

```
.pp
Now is the time for all good men
to come to the aid of their country.
```

will produce a blank line followed by an indented (5 spaces) first line. The result is:

Now is the time for all good men to come to the aid of their country.

PAGE LAYOUT

Spacing

Blank Lines. To insert blank lines in the text where desired, the instruction:

```
.sp N
```

leaves N lines of blank space. When N is omitted, a single blank line is inserted.

Double Spacing. NROFF will automatically double space the output text with the instruction:

```
.ls 2
```

The instruction ".ls 1" will revert back to the single spaced mode. Note: The default is single spacing.

Break Page. Usually, NROFF determines when to start a new page. To start a new page when desired (i.e., not at the default bottom of the page breaking point), the instruction:

```
.bp
```

can be used.

Indention. To indent the left hand margin further than

default, the instruction:

.in +N

can be used, where N is the number of spaces you would additionally like the margin to be indented. To return the margin size to the original number of spaces, use the instruction:

.in -N

Temporary Indention. For a temporary indent, the instruction:

.ti +N

can be used. With the use of this instruction, the indentation applies to the next line only, after which it reverts to the previous amount of indention.

Centering

Text lines can be centered by using the instruction:

.ce N

where N is the following number of lines that are to be centered. When N is omitted, only the line following the ".ce" instruction will be centered horizontally on the page.

HEADERS AND FOOTERS

As desired, headers and footers can be defined, which will be printed at the top and bottom of every page. Within each header or footer, up to three parts may be defined, that is, (1) a left-justified part, (2) a centered part, and (3) a right-justified part. When defining these parts within the headers and footers, a single quote, "'", is used to separate each part. If the user does not desire to define his own headers and footers, the default is the current page number centered at the top of each page.

The format of the header definition macro is:

.he 'left'center'right'

and the format of the footer definition macro is:

```
.fo 'left'center'right'
```

There are several predefined strings of characters that the user may use in the header and footer instructions. These are:

*(dw	-	The day of the week, as a word.
\n(dy	-	The day of the month (numeric).
*(mo	-	The month, as a word.
\n(mo	-	The month of the year (numerical).
\n(yr	-	The last two digits of the current year.
*(td	-	Today's date, in the format month (as a word), day, year.

Some useful examples follow:

1. For a blank header:

```
.he ''''
```

2. For a centered current page number, either:

```
.he ''%''
```

or

```
.fo ''%''
```

Note that the "%" in the above instructions will be replaced with the current page number when it is output.

3. For a centered current page number, the current date (month,day,year) on the right, and "DRAFT" on the left, all at the top of every page:

```
.he 'DRAFT'%'\*(td'
```

This instruction would be output as:

DRAFT -1- February 15, 1982

4. For a centered current page number surrounded by dashes, "FINAL" on the left, and the date (mm/dd/yy) on the left, all at the bottom of every page:

```
.fo 'FINAL'-%-'\n(mo/\n(dy/\n(yr'
```

which would produce:

DISPLAYS

Displays are sections of text that the user desires to be set off from the main body of the document. Depending on the type of display, the user may or may not want the display to be formatted by NROFF. There are several types of displays:

1. Lists
2. Keeps
    a. Blocks
    b. Floating
3. Quotes
4. Centered Blocks

Each display type is discussed in this section.

Lists

A list is a single spaced, unfilled display. Lists should be used when the user does not want the output text to be filled and right-justified. One example of material that might be placed in a list is a column of figures. A list is surrounded by the ".(l" and ".)l" instructions and each list entry is indented 5 spaces. For example:

```
The advantages of this approach are:
.(l
- the more effective use of human resources
- insurance against loss of personnel
- an improved training atmosphere for "junior"
programmers
.)l
```

This input will be formatted as:

The advantages of this approach are:

- the more effective use of human resources
- insurance against loss of personnel
- an improved training atmosphere for "junior" programmers

Keeps

A keep is a display of text which is kept on a single page is possible. Keeps will not be broken over a page boundary (unlike lists). A diagram is an example of the type of text the user would probably put into a keep.

Block Keeps

Blocks are the basic kind of keep. Blocks are surrounded by the instructions ".(b" and ".)b". For example:

```
.(b
This is an example of a block keep.

+-------------------------------------+
| NAME              APPROACH          |
|                                     |
|Top-Down     Based upon decomposition|
|Design        by trial and error     |
|                                     |
|Structured   Based upon decomposition|
|Design        determined by data flow|
|              or data transformations|
+-------------------------------------+
.)b
```

This input will be output as:

This is an example of a block keep.

NAME	APPROACH
Top-Down Design	Based upon decomposition by trial and error
Structured Design	Based upon decomposition determined by data flow or data transformations

If there is not enough space on the current page for the entire block, a new page is started. To avoid leaving large amounts of blank space, the user may choose to split up the block into two smaller blocks or he may choose to use a "floating" keep.

Floating Keeps

Floating keeps move relative to the text. A floating keep will appear at the bottom of the current page if there is enough space; otherwise, it will appear at the top of the next page. Material which will be referred to within the text (e.g., "see figure 1") is often used in a floating keep. Floating keeps are surrounded by the instructions ".(z" and ".)z".

Quotes

Often it is desired to set quotes off from the rest of the text. Those quotes are more indented than the rest of the text and have quote marks around them as the user desires. These quotes are surrounded by the ".(q" and ".)q" instructions. For example, the input:

```
.(q
"Structured programming is the formation of
programs as hierarchical, nested structures
of statements and objects of computation."
.)q
```

generates as output:

> "Structured programming is the formation of programs as hierarchical, nested structures of statements and objects of computation."

Centered Blocks

When the user wants to center several lines as a group, rather than centering them one line at a time, it is convenient to use centered blocks. Centered blocks are surrounded by the instructions ".(c" and ".)c". Centered blocks are not keeps and may be used in conjunction with keeps. By using this facility, all the lines are centered as a unit, such that the longest line is centered and the remaining lines are lined up around that line. For example, the input:

```
.(b
.(c
engineering analysis
simulation
experience data
.)c
.)b
```

will produce:

> engineering analysis
> simulation
> experience data

SPECIAL PARAGRAPHS

This section describes the various types of special paragraphs that the user may want to use in document writing.

Block Style Paragraphs

To generate block style paragraphs (i.e., paragraphs that are left-justified, non-indented), the user may use the ".lp" instruction. For example, the input:

```
.lp
A third useful technique is the use of
software tools, especially text editors,
which will save time and effort.
```

will be output as:

A third useful technique is the use of software tools, especially text editors, which will save time and effort.

Exdented Paragraphs

Exdented paragraphs are those that have the body indented and the first line exdented with a label. These paragraphs are initiated by the ".ip" instruction. For example:

```
.ip one
This is the first paragraph.
The second line will line up with
the first line in this paragraph.
.ip two
This is the second paragraph.
```

This input text will be output as:

one This is the first paragraph. The second line will line up with the first line in this paragraph.

two This is the second paragraph.

The paragraphs generated by the ".ip" instruction are normally indented 5 spaces. If the label is longer than the space allocated for the label, the ".ip" instruction will begin a new line after the label. For example, the input:

```
.ip longlabel
This paragraph has a long label, therefore
the first character of the paragraph itself
will begin on a new line.
```

will generate the output:

longlabel
This paragraph has a long label, therefore the first character of the paragraph itself will begin on a new line.

This instruction may also be used to generate alphabetized paragraphs. Two examples follow:

```
.ip
a.  This is the first item.
Again the second line will be lined
up with the first line.
.ip [b]
This is the second paragraph, which
illustrates a different method.
```

This input will generate:

a. This is the first item. Again the second line will be lined up with the first line.

[b] This is the second paragraph, which illustrates a different method.

Another variation of this instruction is the ability to put longlabels on the paragraphs. This is accomplished by adding a second parameter to the ".ip" instruction. For those users who have used the "-mm" instructions on the DEC 11/70, this variation is similar to the variable list capability. For example:

```
.ip longlabel 10
This paragraph has a long label,
but the "10" parameter will cause
the paragraph to be indented 10 spaces
to allow space for the longlabel.
```

This input will generate the following output:

longlabel This paragraph has a long label, but the "10" parameter will cause the paragraph to be indented 10 spaces to allow space for the longlabel.

Numbered Paragraphs

Automatically numbered paragraphs can be generated by the instruction, ".np". The paragraphs are numbered sequentially from "1" and the numbering is reset at the next ".pp", ".lp", or ".sh" instruction. For example:

```
.np
This is the first paragraph.
.np
This is the second paragraph.
The second line of this paragraph is
also lined up with the other line.
```

This input will be formatted as:

(1) This is the first paragraph.

(2) This is the second paragraph. The second line of this paragraph is also lined up with the other line.

SECTION HEADINGS

Automatically Numbered Headings

Section numbers (e.g., 1.1, 1.2, 1.2.1, etc.) can be automatically generated by using the ".sh" macro instruction. The format of this instruction is:

.sh N "section title"

where "N" is the depth of the section number, that is, how many numbers should be in the section number NROFF will generate. If the user does not want the numbered headings to be underlined, it is necessary to change a register setting at the beginning of the document:

```
.nr sf 3
```

For example:

```
.nr sf 3
.sh 1 "Section 1"
.pp
The text of Section 1.
.sh 2 "Subsection 1"
.sh 2 "Subsection 2"
.sh 1 "Section 2"
This is another major section.
```

This input will generate:

1. Section 1

The text of Section 1.

1.1. Subsection 1

1.2. Subsection 2

2. Section 2 This is another major section.

Unnumbered Headings

Section headers without automatically generated numbers can be produced by using the ".uh" instruction, whose format is:

.uh "section title"

An example of the use of the unnumbered section title is the section header above, "SECTION HEADINGS".

DELAYED TEXT

Delayed text is text that is printed only when it is explicitly called for, such as at the end of each chapter. There are several types of delayed text:

1. Footnotes, printed at the bottom of the current page.

2. References, printed at the end of each chapter or at the end of the document.

3. Indexes, delayed text accompanied by "tags" (usually page numbers).

Footnotes

Footnotes begin with the ".(f" instruction and end with the ".)f" instruction. NROFF automatically maintains the current footnote number and is accessed by the user by typing "**" to produce a footnote number. The number is automatically incremented after every footnote. For example, the input:

```
.(q
There have been several definitions
of software engineering posed since the
early 1970s.\**
.(f
\**Randall W. Jensen
.ul
Software Engineering
Prentice-Hall, Inc., 1979
p.9
.)f
.)q
```

will generate within the document:

> There have been several definitions of software engineering posed since the early 1970s.[1]

---

[1]Randall W. Jensen Software Engineering Prentice-Hall, Inc., 1979 p.9

The footnote itself is appropriately spaced and printed at the bottom of the same page (see bottom of previous page).

Note that the footnote command details appear inside the quote; this insures that the footnote will appear on the same page as the quote.

References

Another type of delayed text is a reference list. References, or delayed text, begin with the ".(d" instruction and end with the ".)d" instruction. As with footnotes, the delayed text instruction appears inside the quote instruction contents. To automatically generate reference numbers, the symbol "*#" is used both at the end of the quote or reference and at the beginning of the reference itself. An example follows:

```
.(q
Software engineering is a discipline, rather than
a technology. \*#
.(d
\*#
.ul
Software Engineering:Problems & Future Development,
J. A. Clapp,
Mitre Corp.,
Prepared for Electronic Systems Division,
Nov.,1974
.)d
.)q
```

This input will produce:

Software engineering is a discipline, rather than a technology.[1]

To print the delayed text (the reference details) when desired, the ".pd" instruction is used. When printed, the reference will appear as follows:

[1] Software Engineering:Problems & Future Development, J. A. Clapp, Mitre Corp., Prepared for Electronic Systems Division, Nov.,1974

Indexes

An index, or table of contents, can be automatically generated by NROFF. The entries that the user chooses to have in the index are automatically saved by NROFF and printed only when called for. Index entries begin with the ".(x" instruction and end with the ".)x" instruction.

The .(x instruction may have a single character argument to specify the name of the index; in this way several indicies may be maintained by NROFF simultaneously to be printed as desired. There are several uses for this ability, such as a list of tables or a list of figures.

Other arguments the .(x instruction may have are:

1. A number, which is the value to print as the page number

2. An underscore, ("_"), which causes no page number or line of dots to be printed.

3. "", an empty pair of quotes, which specifies a null page number.

Without a number argument, the default page number inserted in the index will be the page on which the section appears in the document.

An example of index generation follows:

```
.(x
Introduction
.)x
.(x _
Chapter 1
.)x
.(x 4.2
Chapter 2
.)x
.(x ""
Conclusion
.)x
.xp
```

This input will generate the following output after the ".xp" instruction. Notice that the index must be printed at the end of the paper, rather than at the beginning where it will probably appear in the final product; the pages will have to be physically rearranged after printing.

COLUMNED OUTPUT

NROFF is also able to handle a 2-column mode of output. Additionally, the user is able to turn this mode of output on and off as desired.

The instruction ".2c" tells NROFF to enter the 2-column mode. This mode will be in effect until the ".1c" instruction, to revert to single-column mode, is given. The instruction to begin a new column is ".bc". This instruction begins a new column on a new page only if necessary, rather than forcing a entire new page if there is space for another column on the current page.

When the user calls NROFF to format the input (both text and macro instructions) for 2-column output, he must also call the 2-column post processor, "col". This post processor enables a printer device without a reversable forms feed capability to handle 2-column output. The command for 2-column processing is:

```
nroff -me yourfile|col
```

or

```
nroff -me yourfile|col|cat -d
```

Figure 2 is a sample of 2-column text input. Figure 3 is the resultant output after this sample input has been formatted by NROFF and "col".

```
.ce
WHAT TYPES OF USERS NEED SOFTWARE ENGINEERING?
.sp2
The following sections are intended to serve as
guides to the users of the different computer systems
available at DTNSRDC.
.sp2
.ce
SMALL/ONE-TIME APPLICATIONS
.sp
.2c
Users with small and one-time applications, either
on a strictly interactive system or on a combination
of interactive and batch, should make use of at least
two or three of the Software Engineering techniques
described above.
.pp
The first technique, documentation, provides an
invaluable permanent record of a programmer/scientist's
work.
Often a small task is later added to and becomes a
major effort, or an intended one-time application is
used more than once.
.bc
A complete, permanent description of the initial
effort will make the later task easier and more
time-efficient.
Likewise, it is helpful to have someone not
familiar with the project read your documentation and
try to follow the instructions provided for data
creation and format, input, etc.
Any misdirection discovered at this stage will
save later users many hours of wasted effort.
.1c
.pp
The second technique that will be beneficial
for small/one-time applications is verification.
```

Figure 2 Sample Text for 2-Columned Output

WHAT TYPES OF USERS NEED SOFTWARE ENGINEERING?

The following sections are intended to serve as guides to the users of the different computer systems available at DTNSRDC.

SMALL/ONE-TIME APPLICATIONS

Users with small and one-time applications, either on a strictly interactive system or on a combination of interactive and batch, should make use of at least two or three of the Software Engineering techniques described above.

The first technique, documentation, provides an invaluable permanent record of a programmer/scientist's work. Often a small task is later added to and becomes a major effort, or an intended one-time application is used more than once. A complete, permanent description of the initial effort will make the later task easier and more time-efficient. Likewise, it is helpful to have someone not familiar with the project read your documentation and try to follow the instructions provided for data creation and format, input, etc. Any misdirection discovered at this stage will save later users many hours of wasted effort.

The second technique that will be beneficial for small/one-time applications is verification.

Figure 3 Sample 2-Column Output

OTHER DOCUMENT COMMANDS

Although reports traditionally have the abstract, table of contents, list of figures, etc. at the front of the document, it is more convenient to format and print them last when using NROFF. In this manner, entries (e.g., indexes, figures, and references) are collected throughout the paper and then printed with the correct formats and headers. The following lists several of these useful instructions. The last section of this guideline contains a sample document which illustrates how several of these instructions are used.

.xp n - Print the index; optional parameter, "n", is the index specified by the user. This instruction will print all entries that have been collected in the "n" buffer area by the ".(x n" instruction.

.pd - Print the delayed text. All text that has been saved (collected) via the ".(d" and ".)d" instructions is printed. This may be useful to print references, etc. at the end of each chapter or section, if the user desires.

.++m h - This instruction defines that section of the paper which is being entered. The "m" parameter defines the section type (see the different types below). The "h" parameter defines the new header (not title but header).

Section Types:

.++C - The chapter portion of the paper

.++A - The appendix portion of the paper

.++P - The material following this instruction is the preliminary portion (table of content, list of figures, etc.) of the paper. This instruction also causes the page numbers to be restarted from one in lower case Roman numerals.

.++AB - The abstract portion of the paper follows this instruction.

.++B - The bibliographic portion of the paper, which will be placed at the end, follows this instruction.

Additionally, the ".+c" instruction may be used to define the title for whichever separate section of the paper is being processed. An example of these two types of instructions as they might be used follows:

```
.++B
.+cBIBLIOGRAPHY
 ...text of bibliography...
.++P
.bp
.ce
TABLE OF CONTENTS
 ...table of contents...
```

SAMPLE NROFF DOCUMENT

The following text is a sample document, which is ready to be input to NROFF. This sample incorporates many of the macro instructions, rules, and conventions discussed in this guideline which are necessary to produce NROFF formatted text.

Note that the page numbers for this sample document are numbered consistent with this main document, rather than beginning with "1", as they would normally be numbered in a separate document. Additionally, note that the Table of Contents page that is produced for this sample document has the Roman numeral "i" as its page number.

```
.he ''''
.fo 'DRAFT'-%-' \*(td'
.ce
ABSTRACT
.(x
ABSTRACT
.)x
.sp
.pp
This report gives users of the DTNSRDC computers
an overview of the principles of software engineering
and describes various software engineering techniques
that users with either large or small applications would
find beneficial.
.sp2
.ce
BACKGROUND & DEFINITION
.(x
BACKGROUND & DEFINITION
.)x
.sp
.pp
The term "Software Engineering" was coined during the
late 1960's at a NATO Science Committee conference.
Since then, many articles and books have been
published in an attempt to define this approach
to producing and managing software.
.sp
.(q
"Software engineering is not just a collection of tools
and techniques, it is ... a full-fledged engineering
discipline ...". \*#
.(d
\*#  Jensen, R. W. and C. C. Tonies,
.ul
Software Engineering,
Prentice-Hall, Inc.
(1979).
.)d
.)q
.(q
"... the establishment and use of sound
engineering principles (methods)
in order to obtain economically software that
is reliable and works on real machines." \*#
.(d
\*#  Bauer, F. L.,
"Software Engineering,"
.ul
Information Processing 71,
Amsterdam:North Holland Publishing Co.,
p. 530,
1972
```

```
.)d
.)q
.sp2
.ce
PURPOSE
.(x
PURPOSE
.)x
.sp
.pp
High-quality software can be defined as having
the following characteristics:
.(b

  1. Reliability                    4. Portability
  2. Clarity                        5. Efficiency/Economy
  3. Maintainability                6. Testability


.)b
The advantages of this approach are:
.ip a.
the more effective use of human resources
.ip b.
a better design
.ip c.
insurance against loss of personnel
.bp
.ce
REFERENCES
.(x
REFERENCES
.)x
.sp3
.pd
.++P
.bp
.ce
TABLE OF CONTENTS
.sp3
.ls 2
.xp
```

The following 3 pages of text is the formatted output which resulted from putting the previous sample text through NROFF (and the post processor, "col").

ABSTRACT

This report gives users of the DTNSRDC computers an overview of the principles of software engineering and describes various software engineering techniques that users with either large or small applications would find beneficial.

BACKGROUND & DEFINITION

The term "Software Engineering" was coined during the late 1960's at a NATO Science Committee conference. Since then, many articles and books have been published in an attempt to define this approach to producing and managing software.

> "Software engineering is not just a collection of tools and techniques, it is ... a full-fledged engineering discipline ...".[1]

> "... the establishment and use of sound engineering principles (methods) in order to obtain economically software that is reliable and works on real machines."[2]

PURPOSE

High-quality software can be defined as having the following characteristics:

1. Reliability
2. Clarity
3. Maintainability
4. Portability
5. Efficiency/Economy
6. Testability

The advantages of this approach are:

a. the more effective use of human resources

b. a better design

c. insurance against loss of personnel

REFERENCES

[1] Jensen, R. W. and C. C. Tonies, Software Engineering, Prentice-Hall, Inc. (1979).

[2] Bauer, F. L., "Software Engineering," Information Processing 71, Amsterdam:North Holland Publishing Co., p. 530, 1972

TABLE OF CONTENTS

INITIAL DISTRIBUTION

COPIES:

12 DIRECTOR
DEFENSE TECHNICAL INFORMATION CENTER (DTIC)

CENTER DISTRIBUTION

COPIES:

1	18/1809	GLEISSNER, G. H.
1	1804	AVRUNIN, L.
1	1805	CUTHILL, E. H.
2	1809.3	HARRIS, D.
1	182	CAMARA, A. W.
1	184	SCHOT, J. W.
1	185	CORIN, T.
1	187	ZUBKOFF, M. J.
1	189	GRAY, G. R.
100	189.1	HIBBERT, D. J.
1	189.2	HAYDEN, H. P.
1	189.3	COOPER, A. E.
10	1892.1	STRICKLAND, J. D.
1	1892.2	SOMMER, D. V.
1	1892.3	MINOR, L. R.
1	1894	SEALS, W.
1	1896	GLOVER, A.
1	1896.2	DENNIS, L.
1	522	LIBRARY, CARDEROCK
1	522.2	LIBRARY, ANNAPOLIS
1	93	OFFICE OF PATENT COUNSEL

DTNSRDC ISSUES THREE TYPES OF REPORTS

1. DTNSRDC REPORTS, A FORMAL SERIES, CONTAIN INFORMATION OF PERMANENT TECHNICAL VALUE. THEY CARRY A CONSECUTIVE NUMERICAL IDENTIFICATION REGARDLESS OF THEIR CLASSIFICATION OR THE ORIGINATING DEPARTMENT.

2. DEPARTMENTAL REPORTS, A SEMIFORMAL SERIES, CONTAIN INFORMATION OF A PRELIMINARY, TEMPORARY, OR PROPRIETARY NATURE OR OF LIMITED INTEREST OR SIGNIFICANCE. THEY CARRY A DEPARTMENTAL ALPHANUMERICAL IDENTIFICATION.

3. TECHNICAL MEMORANDA, AN INFORMAL SERIES, CONTAIN TECHNICAL DOCUMENTATION OF LIMITED USE AND INTEREST. THEY ARE PRIMARILY WORKING PAPERS INTENDED FOR INTERNAL USE. THEY CARRY AN IDENTIFYING NUMBER WHICH INDICATES THEIR TYPE AND THE NUMERICAL CODE OF THE ORIGINATING DEPARTMENT. ANY DISTRIBUTION OUTSIDE DTNSRDC MUST BE APPROVED BY THE HEAD OF THE ORIGINATING DEPARTMENT ON A CASE-BY-CASE BASIS.