Data General

Customer Documentation

Documenter's Tool Kit Technical Summary for the DG/UX™ System

AViiON™
PRODUCT LINE

Documenter's Tool Kit Technical Summary for the DG/UX™ System

069-701041-00

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.

Ordering No. 069-701041
All Rights Reserved
Printed in the United States of America
Revision 00, May 1989

NOTICE

DATA GENERAL CORPORATION (DGC) HAS PREPARED AND/OR HAS DISTRIBUTED THIS DOCUMENT FOR USE BY DGC PERSONNEL, LICENSEES, AND CUSTOMERS. THE INFORMATION CONTAINED HEREIN IS THE PROPERTY OF THE COPYRIGHT HOLDER(S); AND THE CONTENTS OF THIS MANUAL SHALL NOT BE REPRODUCED IN WHOLE OR IN PART NOR USED OTHER THAN AS ALLOWED IN THE APPLICABLE LICENSE AGREEMENT.

The copyright holders reserve the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF DGC HARDWARE PRODUCTS AND THE LICENSING OF DGC SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN DGC AND ITS CUSTOMERS, AND THE TERMS AND CONDITIONS GOVERNING THE LICENSING OF THIRD PARTY SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE APPLICABLE LICENSE AGREEMENT. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY DGC FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF DGC WHATSOEVER.

IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS DOCUMENT OR THE INFORMATION CONTAINED IN IT, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES.

UNIX is a U.S. registered trademark of AT&T.
DG/UX is a trademark of Data General Corporation.

Copyright © AT&T 1988
Copyright © Data General Corporation 1988
All Rights Reserved
Printed in the United States of America

Certain portions of this document were prepared by Data General Corporation, and the remaining portions were prepared by AT&T.

Documenter's Tool Kit Technical Summary for the DG/UX™ System
069-701041-00

Effective with DG/UX, Revision 4.10

Documenter's Tool Kit Technical Summary for the DG/UX System

Introduction

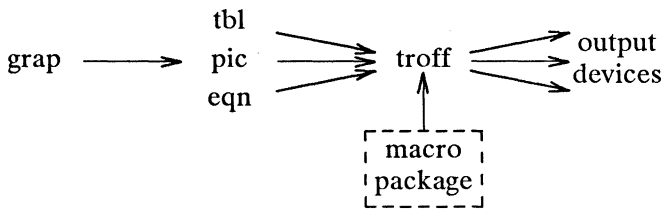
The Documenter's Tool Kit Software provides a family of programs for typesetting materials containing equations, tables, and diagrams, as well as standard text. The base upon which these interrelated programs rest is a text formatter called **troff**, a generic term for **nroff/troff**, developed by Joseph P. Ossanna around 1973. By itself, **troff**, with its many requests and escape sequences, provides a highly detailed level of text processing, performing few high-level formatting operations beyond line filling and justification. It accepts input from other, more specialized programs. In particular, page layout is controlled by the **mm** macro package, so-called because **troff** is a macro processor.

Several Documenter's Tool Kit Software programs deal with specific areas of document preparation. **eqn**, for example, is a language for typesetting mathematical expressions. It acts as a preprocessor for **troff**. That is, the high-level instructions you give **eqn** are compiled into the lower-level **troff** code that will actually plot the desired mathematical notation. The other Documenter's Tool Kit Software preprocessors are **eqn**, **tbl**, **pic**, and **grap**.

grap, actually, is a "pre-preprocessor." This language, which you use to make graphs of numerical data, accepts English-oriented input and directs its output to **pic**, which in turn feeds **troff**.

Finally, output, however produced, can be directed to any of a large number of devices, ranging from conventional typesetters to laser printers and bitmap terminals.

The figure shown below gives an overview of this input-output activity:



In addition, the figure suggests the characteristic usage of Documenter's Tool Kit tools. Rather than one program doing all of the work, several specialized programs accomplish a wide variety of diverse functions. This approach enables each component to use an individually tailored language particularly suited to its own purposes. That is, Documenter's Tool Kit Software is not top-heavy with a single, lowest-common-denominator language that must address a multitude of different demands. Rather, each program's "dialect" is dedicated, relatively stream-lined, and easy to use.

troff

troff traces its origins back to a group of formatters that developed from RUNOFF by J. E. Saltzer at MIT during the 1960s. (The name *roff* actually derives from a primitive formatter that Brian Kernighan wrote for his own use as a graduate student.) **troff**'s name was coined to express *typesetter run-off*. It uses formatting commands, called requests, that are interspersed through the text to govern processing. A report title, for example, might be formed with the following two lines of requests:

```
.ce 1
.ft B
```

which would place the next line of text in a centered position and change the font to bold. **troff** provides eighty-five such requests, which are in turn effectively multiplied by their respective numerical and alphabetical arguments.

In addition, **troff** uses about forty in-line commands for changing character size and typeface, placing special characters, invoking previously defined characters or words, and dictating a variety of movements about the page, to name a random assortment. For example, **\fI** changes all succeeding text to italic, **\(co** gives the encircled copyright symbol, and **\h***'numeric argument'* moves text a distance indicated by a simple numeric argument, arithmetic expression, register evaluation, or a combination of these. Arguments concerning physical dimensions may specify inches, centimeters, picas, ems, ens, points, and machine units.

troff, unlike most of its its related components, does not have individual requests or escape sequences for automatically providing running titles at the top of the page, setting up a table of contents, or making concordances. In contrast to say, the **mm** macro package, it is intended to permit the user a detailed access to the inner workings of Documenter's Tool Kit Software.

troff is programmable; it has variables, arithmetic arguments, and conditional tests of many things, including the dimensions of processed text. It allows you to define macros that encapsulate a sequence of operations or that receive and store processed text. It also permits you to set traps (marked page positions) that pass control to macros you have set up for this purpose.

While this advanced level of detailed access can be indispensable, it is not always desirable. Higher-level components, such as **mm** macros, should be used for most text formatting. **troff** should be thought of as the instrument for fine-tuning that enables you to accomplish the non-standard processing that more generalized components do not offer. Furthermore, **troff** should be understood as a language especially designed for text programming. It is capable of sophisticated manipulation of typography, evaluation of accumulated text (or "diversions"), placement of graphical elements contingent upon diversion evaluation, and detailed page control.

Macro Packages

Documenter's Tool Kit Software uses a number of macro packages that enable the user to think in terms of text—title, page headings, footnotes—rather than in terms of typography. Each of these packages is designed to accomplish specialized tasks ranging from memorandum and report preparation to compiling indexes.

The **mm** macro package is by far the most powerful and varied of these. Formed as a library of **troff** request and string combinations, these macros also allow for limited programmable features. **mm** provides the tools you would need for most text processing: static or floating displays, seven levels of numbered page headings, one- or two-column formatting, table-of-contents formatter (collected automatically from numbered headings), to name a few features. Many of these standard features can be altered to suit individual tastes.

Documenter's Tool Kit Software also includes the **mptx** macro package, used to make permuted indices and the **mv** macro package for making view graphs and projection slides. Documenter's Tool Kit Software indexing tools are **mptx**, **ndx**, **ptx**, and **subj**.

Preprocessors

eqn defines a simple language for representing mathematical expressions. People trained in mathematics can usually learn it in a few minutes; people without such training can normally use the language effectively in an hour or so. For example:

```
a+b over c+d+e = -b sup 2 over pi
```

generates this:

$$\frac{a+b}{c+d+e}=\frac{-b^2}{\pi}$$

Because **eqn** runs as a preprocessor, the entire document that contains the equation is passed through it before reaching **troff**, which accepts **eqn**'s output as input. **eqn** does not, however, influence non-mathematical parts of the text. It looks for language native to **eqn** (usually placed between the macros **.EQ** and **.EN**) and ignores the rest.

tbl is **troff**'s preprocessor for making tables. It uses a language altogether different from the other preprocessors, which are more oriented toward spoken English. It is, nonetheless, a simple language to learn and can usually be mastered after one session. The following input lines are an example:

```
.TS
center, doublebox;
cfB s s
c c c
^ c c
lp8 n n.
Program Sizes
Name      Source    Object bytes
          Lines     (text+data)
_
\f3troff\fP         8681      73136
\f3eqn\fP           1821      34164
\f3tbl\fP           2581      39936
\f3pic\fP           3760      83968
\f3grap\fP          2791      58368
.TE
```

(The columns of listed data items are separated by tabs.)

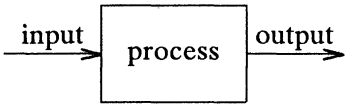
The output looks like this:

Program Sizes		
Name	Source Lines	Object bytes (text+data)
troff	8681	73136
eqn	1821	34164
tbl	2581	39936
pic	3760	83968
grap	2791	58368

pic is the facility for drawing figures and diagrams. It places forms (boxes, circles, lines, and splines) at absolute positions (using Cartesian coordinates) or at specified positions relative to already placed objects. The following is an example of **pic** input:

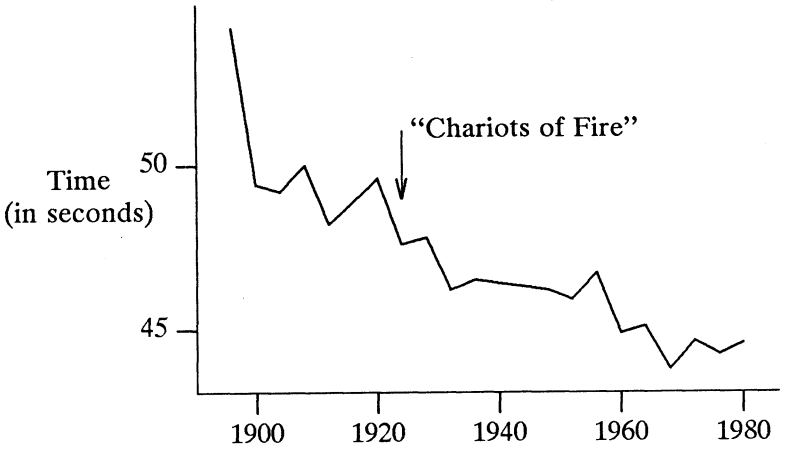
```
.PS
arrow "input" above
box "process"
arrow "output" above
.PE
```

and it will produce the following illustration:



You may follow each object with attributes that control its size and position and associated text strings. You may, in addition, plot forms other than the ones that **pic** already supplies. A macro facility allows common operations to be encapsulated, and these macros will accept arguments, enabling you to build a library of **pic** drawings which, furthermore, can be modified at each usage.

grap provides a language for describing graphs. Its output is **pic** input. **pic** in turn feeds requests and strings to **troff**. By default **grap** plots input numbers as a scatter plot, with automatically computed tick marks. Other commands provide control over labels, scaling, logarithmic coordinates, and the like. The following is a **grap** graph:



Olympic 400 Meter Run: Winning Times

which was produced by the following input:

```
.G1
frame top invis right invis
label left "Time" "(in seconds)"
label bot "Olympic 400 Meter Run: Winning Times"
draw solid
1896 54.2
1900 49.4
1904 49.2
1908 50.0
. . .
1980 44.60
arrow from 1924,51 to 1924,49
" ``Chariots of Fire''" ljust at 1924,51
.G2
```

grap, like its near-relative **pic**, has a macro facility and control-flow features.

mm: Technical Discussion

1. Introduction

This is a technical discussion of the **mm** (Memorandum Macro) package, a collection of macros that you use to format documents such as letters, reports, memoranda, papers, manuals, and books. Macros are formatting commands that combine the functions of one or more **nroff** or **troff** requests.

Read this technical discussion if you have experience using the **mm** package. If you have never used **mm** before, you first might want to read "The Macro Package **mm**: a Tutorial" in *Using the Documenter's Tool Kit on the DG/UX System*. Once you decide what macros you want to use, refer to this discussion for details about how to use them.

1.1. How this Technical Discussion is Organized

This first chapter offers definitions of terms that are basic to text processing, specifies conventions that this technical discussion uses, and describes formatting concepts relevant to your use of **mm**. It is provided to enrich and clarify the discussion of each facility of **mm** that follows.

Assume that a document that you format with **mm** consists of four segments: a parameter setting segment, a beginning, a body, and an end. "Chapter 2" discusses these segments, which provide the organizing principle for the rest of this technical discussion. "Chapter 3" presents macros relevant to formatting the body of a document. In "Chapter 4," macros for formatting the beginning of a document are described. "Chapter 5" explains end macros. Finally, "Chapter 6" covers miscellaneous macros. In each of these chapters, a macro's default formatting action is described followed by ways to change this default behavior. For the most part, macros are described starting with the simplest usage and progressing to more advanced cases. It might be best to read a section in detail only to the point where you have enough information to obtain the result that you want and then to skim the rest.

After covering all **mm** macros, this technical discussion tells you about troubleshooting and error messages in "Chapter 7." "Chapter 8" gives a full account of using the **mm** command line to process the files you create. Appendices are presented in "Chapter 9."

1.2. Definitions

"Formatter" refers to either the **nroff** or the **troff** text-formatting program. These programs have their own tutorials in *Using the Documenter's Tool Kit on the DG/UX System*. and their own technical discussion in this book. Unless a functional distinction requires otherwise, "the formatter" refers to both **nroff** and **troff**.

Requests are built-in commands recognized by the formatter. Although you seldom need to use these requests directly when you use **mm**, this section contains references to some requests. For example, the request **.sp** inserts a blank line in the output at the place that the request occurs in the input file. Request names consist of two lower-case letters. For an introduction to **nroff/troff**, see "The Formatter **nroff**," or "The Formatter **troff**" in *Using the Documenter's Tool Kit on the DG/UX System*. For details, refer to the "**nroff/troff** Technical Discussion" in this book.

Macros are named collections of requests. That is, each macro is an abbreviation for a collection of requests that are used repeatedly in the same combination. Rather than typing them each time they are needed, you simply type the macro that calls them. Sometimes, macros are composed of strings and number registers, which are described below.

The **mm** package supplies many macros, and you can define additional ones. The names of **mm** macros consist of one or two upper-case letters, so if you create your own formatting macros, name them with a lower-case and upper-case letter, for example, **.pD**, to prevent usurping the function of an **mm** macro. Appendix A, the "**mm** Macro Name Summary," gives a complete listing of **mm** macros.

Strings are character variables, each of which names a string of alpha-numeric characters. Page headers, page footers, and lists often contain strings. You give a string a value with the **.ds** (define string) request, and you obtain its value by typing its name, preceded by "*****" (for 1-character names) or "***(**" (for 2-character names). For instance, the string **DT** in **mm** normally contains the current date, thus the input line

```
Today is \*(DT.
```

results in output such as this:

```
Today is February 3, 1989.
```

You can replace the current date by redefining the contents of this string, for example,

```
.ds DT 06/04/85
```

or by invoking a macro designed for that purpose {4.1.3}. Appendix B provides the "**mm** String Name Summary."

Number registers, which are similar to integer variables, store the value of various text parameters, such as the number of spaces between paragraphs (register **Ps**). The formatter program uses these registers for flags, for arithmetic, and for automatic numbering. You can give certain registers a value using the **.nr** request, and you can reference them by preceding their names by **\n** (for 1-character names) or **\n(** (for 2-character names, for example, **\n(Ps**).

The following line creates a new number register **d**, and sets its value to one more than that of another new register **dd**: **.nr d 1+\n(dd**. Some number registers are read-only. That is, you can reference them, but you cannot change their value. An "**mm** Number Register Summary" appears as Appendix C to this technical discussion.

1.3. Conventions

Numbers enclosed in curly braces ({ }) refer to section numbers. For example, this is section {1.3}.

In the synopses of macro calls, square brackets ([ ]) surrounding an argument show that it is optional. An argument in italics means that you are to substitute a legal value for that argument. Ellipses (. . .) show that the preceding argument may appear more than once.

In cases where the behavior of the two formatters **nroff** and **troff** is obviously different, the **nroff** formatter output is described first with the **troff** formatter output following in parentheses. For instance,

> The title is underlined (italic).

means that the title is underlined by the **nroff** formatter and italicized by the **troff** formatter.

1.4. Formatting Concepts

1.4.1. Basic Terms

A formatter normally fills output lines from one or more input lines. You may justify output lines so that both the left and right margins are aligned. As you fill lines, you may also hyphenate words as necessary {1.4.4}. It is possible to turn any of these modes on and off (use **.SA** {6.2}, **Hy** {1.4.4}, and the **.nf** and **.fi** formatter requests). Turning off line filling also turns off justification and hyphenation.

Certain requests and macros cease line filling, print the input line (of whatever length), and begin a new output line after the printed text. This printing of a partially filled output line is known as a line break. A few formatter requests and most of the **mm** macros cause a line break.

You can use formatter requests {1.4.10} with **mm**, but there are consequences and side effects that each such request might have. Generally, you should use **mm** macros alone because

- They are much easier to use, to control, and later to change the overall style of the document.
- You obtain complex features (such as footnotes or tables of contents) with ease.
- You are freed from having to define many page control functions in the **nroff** or **troff** languages.

1.4.2. Arguments and Double Quotes

For any macro call, a null argument is an argument whose width is zero. The preferred form for a null argument, which often has a special meaning, is "". Omitting an argument is not the same as supplying a null argument (for example, see the **.MT** macro {4.1.1}). You can omit arguments only at the end of an macro argument list, but you can place null arguments anywhere in the list.

Enclose any macro argument containing ordinary (paddable) spaces in double quotes, or **mm** will interpret the spaces as argument delimiters. A double quote (") is a single character that must not be confused with two apostrophes (''), two acute accents (´´), or two grave accents (``).

You may not use double quotes as part of the value of a macro argument or of a string that you use as a macro argument. If you must have double quotes in a macro argument value, use two grave accents (``) or two acute accents (´´) instead. This restriction is necessary because many macro arguments are processed a variable number of times.

1.4.3. Unpaddable Spaces

When the formatter justifies output lines to give an even right margin, it may append additional spaces to existing spaces in a line. This may distort the desired alignment of text. To avoid this distortion, you must specify a space that cannot be expanded during justification. There are two ways to accomplish this:

- You may type a backslash followed by a space (\ ). These characters directly generate an unpaddable space.
- You may use some seldom-used character to be translated into a space on output.

Because this translation occurs after justification, you may use the chosen character anywhere an unpaddable space is desired. The tilde (~) is often used with the translation request for this purpose. To use the tilde in this way, put the following request at the beginning of your document: **.tr ~**, where the tilde is followed by a space. If you must put a tilde in the output, you can temporarily "recover" it by inserting **.tr ~~** before the place where you need it. Repeating the **.tr ~** restores its usage as a space after a line break or after the line containing the tilde has been flushed from the line buffer.

You should not translate the tilde character into a space when you use it within the **.EQ** and **.EN** macros or assigned **eqn** delimiters.

1.4.4. Hyphenation

Formatters do not hyphenate unless you request it. You can turn on hyphenation in the body of the text by typing the following request at the beginning of the input file: **.nr Hy 1**. Section 3.3.1 describes hyphenation used within footnotes and across page boundaries.

If you request hyphenation, the formatters will automatically hyphenate words as necessary. However, you may specify hyphenation points for a specific occurrence of any word with a special character known as a hyphenation indicator, or you may specify hyphenation points for a small list of words (about 128 characters). If the hyphenation indicator (initially, the 2-character sequence "**\%**") appears at the beginning of a word, the word is not hyphenated. Alternatively, you can use the indicator to show legal hyphenation points inside a word. All occurrences of the hyphenation indicator disappear on output.

You may specify a different hyphenation indicator. The circumflex (^) is often used for this purpose by inserting the following macro at the beginning of a document input text file: **.HC ^**. Any word containing hyphens or dashes (also known as em dashes) is hyphenated immediately after a hyphen or dash if hyphenation is necessary, even if the hyphenation function is turned off.

You may supply (via the exception word **.hw** request) a small list of words with the proper hyphenation points shown. For example, to show the proper hyphenation of the word "printout," you may specify **.hw print-out**.

1.4.5. Tabs

The macros **.MT** {4.1.1}, **.TC** {5.4}, and **.CS** {5.6} use the formatter tabs **.ta** request to set tab stops. The default values of tab settings are every eight characters in the **nroff** formatter, and every ½ inch in the **troff** formatter. You may set tabs to other values.

For the **nroff** formatter, default tab setting values are 8, 16, 24, 32, 40, ..., 160 characters for a total of 20 tab stops. That is, the default tab settings correspond to the following example:

```
.ta 8 16 24 32 40 48 56 64 72 . . . 160
```

You may separate tab settings with commas, spaces, or any other non-numeric character. You may set tab stops in any horizontally oriented scale.

The formatter interprets a tab character with respect to its position on the input line rather than its position on the output line. In general, you should only put tab characters on lines after you turn off line filling (**.nf**) {1.4.10}. The **tbl** program {3.7} changes tab stops but does not restore default tab settings.

1.4.6. Bullets

The **troff** formatter provides the bullet character (•). For compatibility with **troff**, **mm** also provides a bullet string: ***(BU**. The bullet list (**.BL**) macro {3.2.5} uses this string to generate automatically the bullets for bullet listed items.

1.4.7. Dashes, Minus Signs, and Hyphens

The **troff** formatter distinguishes among the dash, the minus sign, and the hyphen, but the **nroff** formatter does not.

- If you intend to use **nroff**, you may only use the minus sign (–) for the minus, hyphen, and dash.
- If you plan to use **troff** primarily, you should follow **troff** escape conventions.
- If you plan to use both formatters, take care during input text file preparation. Unfortunately, these graphic characters cannot be represented in a way that is both compatible and convenient for both formatters.

The following approach is suggested:

Dash — Type "*(EM" for each text dash for both **nroff** and **troff** formatters. This string generates an em dash in the **troff** formatter and two dashes (--) in the **nroff** formatter. Dash list (**.DL**) macros {3.2.5} automatically generate the em dash for each list item.

Hyphen — Type "-" and use as is for both formatters. The **nroff** formatter will print it as is. The **troff** formatter will print - (a true hyphen).

Minus — Type "\-" for a true minus sign regardless of formatter. The **nroff** formatter will ignore the ****. The **troff** formatter will print – (a true minus sign).

1.4.8. Trademark String

A trademark string ***(Tm** is available with **mm**. This places the letters "TM" one-half line above the text that it follows. For example,

```
Using the
.I
DG/UX\f1\*(Tm
System
.R
is available from the library.
```

yields

Using the
DG/UX™
System
is available from the library.

1.4.9. BEL Character

Many macros use the non-printing character BEL as a delimiter to compute the width of an argument or to delimit arbitrary text in page headers and footers {3.4}, headings {3.5}, and lists {3.2}. This is to decrease the possibility that an argument might contain a character identical to one delimiting it, which would produce undesirable results.

1.4.10. Use of Formatter Requests

You need not use most formatter requests with **mm** since it provides the corresponding formatting functions in a more straightforward fashion. The following requests can be useful with **mm**:

.af	Assign format
.br	Break
.ce	Center
.de	Define macro
.ds	Define string
.fi	Fill output lines
.ft	Change font
.hw	Exception word
.ls	Line spacing
.nf	No filling of output lines
.nr	Define and set number register
.nx	Go to next file (does not return)
.rm	Remove macro or string
.rr	Remove register
.rs	Restore horizontal spacing
.so	Switch to source file and return
.sp	Space
.ta	Tab stop settings
.ti	Temporary indent
.tl	Title
.tr	Translate
.sy	Issue command(s) to DG/UX system

The **.fp**, **.lg**, and **.ss** requests are sometimes useful for the **troff** formatter. In general, it is best not to use too many **troff** requests in conjunction with **mm**.

2. The Structure of a Document

A document that you format with **mm** consists of four segments, any of which you may omit. If you include any of these segments, you must put them in the following order:

- The parameter setting segment sets the general style and appearance of a document. Here, you control page length and width, margin justification, numbering styles for heading and lists, page headers and footers, proprietary markings, among other properties of the document. Also, you can add macros or redefine existing ones. You can omit this segment entirely if you are satisfied with **mm**'s default values; the segment produces no output but performs only the formatter setup for the rest of the document. Here is an example of a parameter setting segment:

  .nr Ls 0 specifies that no spacing occurs between any list items

  .nr Cl 4 saves up to 4th level section headings for table of contents

  .nr Pi 7 sets paragraph indentation to 7 spaces

  .nr Hs 4 specifies a line of space between text and section headings up to 4th level

- The beginning consists of those items that occur only once at the start of a document: a memorandum title, names, the date, and so on. Here is the beginning of a formal memorandum:

  .PM PM3 specifies a proprietary marking of "SEE PROPRIETARY NOTICE ON COVER PAGE"

  .TL specifies that line following the macros, "Work Report," is the title

  .AU "J. Smith" JS formats information about the author, "J. Smith"

  .MT **sets the formal memorandum type**

Here is the beginning of a business letter:

.WA signals the beginning of the writer's address

.WE signals the end of the writer's address

.IA signals the beginning of the addressee's address

.IE signals the end of the addressee's address

.LO SA "Dear Jane,"
sets the letter's salutation to "Dear Jane,"

.LT BLspecifies a Blocked type business letter

- The body of a document is the text itself. It may be as little text as a single paragraph or as much as hundreds of pages. It may have a hierarchy of section headings up to seven levels deep {3.5}, and you may automatically number section headings and save them to generate the table of contents. **mm** provides five additional levels of subordination by a set of list macros for automatic numbering, alphabetic sequencing, and "marking" of list items {3.2.1}. You can put various types of displays {3.6}, tables {3.7}, figures {3.9}, equations {3.8}, references {5.7}, and footnotes {3.3} in the body.
- The end contains items that usually occur at the close of a document. Included are signature(s), and lists of notations (for example, "Copy to" lists) {5.3}, which may occur at the beginning of the document {4.2.1}.) You may call certain macros at the end to print information that is wholly or partially derived from the rest of the document such as the table of contents or the cover sheet {5.6}.

For example,

.FC prints the formal closing, "Yours very truly,"

.SG prints the name(s) specified with **.AU**

.NS begins a "Copy to" list

.NE signals the end of the "Copy to" list

.TC generates a table of contents

The existence and size of these four segments varies widely among different document styles. Although a specific item of a segment (such as date, title, author names, and so on) may differ depending on the document, there is a uniform way of typing it into an input text file. To make it easy to edit or revise input file text at a later time:

- Keep input lines short.

- Break lines at the end of clauses.
- Begin each new sentence on a new line.

3. Formatting the Body of a Document

3.1. Formatting Paragraphs (.P)

To use **.P**, which stands for paragraph, your input would look like this:

```
.P [type]
Text
```

.P without an argument forces left justification (the first line begins at the left margin), as does **.P 0**. **.P 1** indents the first line five spaces unless you specify another amount of indentation by changing the value contained in the number register **Pi**. For example, to indent particular paragraphs ten spaces, type the following line once at the top of your file: **.nr Pi 10** and then use **.P 1** before every paragraph that you want indented.

3.1.1. Paragraph Type (Pt)

Suppose that you want all paragraphs indented. Rather than type **.P 1** , you can set the **Pt** number register, which controls the paragraph type. The initial value of **Pt** is 0, which provides left-justified paragraphs. Force every paragraph in your output to be indented by inserting the following line at the beginning of the document input file: **.nr Pt 1**. You may specify the amount of indentation by setting **Pi** as before if you do not want to use the default.

Indent all paragraphs except after headings, lists, and displays (discussed below) by entering the following at the beginning of your document input file: **.nr Pt 2.**

Both the **Pi** and **Pt** register values must be greater than zero to indent paragraphs. Values that you use to specify indentation must be unscaled and are treated as character positions (ens). **nroff** understands an en to be equal to the width of a character. **troff** understands an en to be the number of points (1 point = 1/72 of an inch) equal to half the current point size.

Regardless of the value of **Pt**, **.P 1** causes indentation by the amount specified by the register **Pi**. If **.P** occurs inside a list, the indent (if any) of the paragraph is added to the current list indent {3.2.1}.

3.1.2. Numbered Paragraphs (Np)

Produce numbered paragraphs by setting the **Np** register to 1, which numbers paragraphs within first level headings. Use the **.nP** macro rather than the **.P** macro to produces paragraphs that are numbered within second level headings.

```
.H 1 "FIRST HEADING"
.H 2 "Second Heading"
.nP
These numbered paragraphs contain a "double-line indent,"
in which the text of the second line aligns with the text
of the first line, so that the number stands out.
The third and following lines of a numbered paragraph return
to the left margin.
```

produces

```
                                - 1 -


1.  FIRST HEADING

1.1  Second Heading

1.01  These numbered paragraphs contain a ``double-line
indent,''
 in which the text of the second line aligns with the text of
 the first line, so that the number stands out. The third and
 following lines of a numbered paragraph return to the left
 margin.
```

3.1.3. Spacing Between Paragraphs (Ps)

The **Ps** number register controls the amount of spacing between paragraphs. By default, the formatter sets **Ps** to 1, yielding one blank space (one-half vertical space). Giving **Ps** a value of 0 yields no space between paragraphs.

3.2. Formatting Lists

3.2.1. General Characteristics of Lists

mm provides a convenient way to create lists automatically. All lists are composed of three basic parts:

- A list-initialization macro determines the line spacing, indentation, marking with special symbols, and numbering or alphabetizing of list items. Available list-initialization macros are

  .AL Automatically Incremented List

  .ML Marked List

  .VL Variable-Item List

  .BL Bullet List

  .DL Dash List

  .RL Reference List

  If you do not provide arguments to the list-initialization macro, text will be indented by a default number of spaces from the indent currently in force. This default varies as a function of what type of list you call. Change the indentation value by placing the desired indentation into the number register **Li**.

- One or more list-item macros (**.LI**) identifies each item in your list. List-item macros are followed by the text of the corresponding list item:

  .LI [*mark* [1]]
  Text

Use the **.LI** macro with all list types and for each list item. **.LI** normally causes output of a single blank line before its list item although you may suppress this feature by setting the **Ls** (list space) register. **Ls** is set to the innermost list level in nested lists for which spacing is done. For example, **.nr Ls 0** specifies that no spacing will occur around any list items. The default value for **Ls** is 6 (which is the maximum list nesting level).

You may supply arguments to **.LI**.

- If you give **.LI** no arguments, it labels the item with the *mark* specified by the most recent list-initialization macro (for example, **.BL** sets the mark to be a bullet).
- If you give **.LI** a single argument, that argument is output instead of the current *mark*.
- If you give **.LI** two arguments, the first argument becomes a prefix to the current *mark*, allowing you to emphasize one or more items in a list.

For example,

```
.BL
.LI
This is a bullet item.
.LI +
This replaces the bullet with a "plus."
.LI + 1
This uses a "plus" as prefix to the bullet.
.LE
```

when formatted yields

```
                        - 1 -


 • This is a bullet item.

 + This replaces the bullet with a "plus."

+ • This uses a "plus" as prefix to the bullet.
```

Do not put ordinary (paddable) spaces into the *mark* because the alignment of items is lost if you justify the right margin {1.4.3}.

If the current mark in the current list is a null string, and the first argument of **.LI** is omitted or null, the resulting effect is that of a hanging indent. That is, the first line of the following text is "outdented," starting at the same place where the mark would have started {3.2.4}. The list-end macro (**.LE**) ends the list: **.LE** [**1**]. If you specify an argument to **.LE**, it outputs a blank line.

The list-initialization macro saves the previous list status (indentation marking style, and so on), and changes the status to that of the new initialization macro. The list-end macro restores the status of the previous list unless there is no previous list. In that case, the list-end macro restores the status to that existing before the list-initialization macro call. This information about list status is important to remember when you format nested lists, which are described below.

3.2.2. Automatically Incremented Lists (.AL)

.AL stands for automatically incremented list. The general syntax of the macro is as follows: **.AL** [*type* [*text-indent* [1]]].

If you do not specify arguments, the list is numbered, and the text is indented the value of **Li**, initially six (five) spaces from the indent in force when the **.AL** is called, leaving room for a space, two digits, a period, and two spaces before the text.

Do not scale values that specify indentation. These values are scaled in terms of "character positions" (ens).

Specifying a *type* produces a different type of sequencing. The value of *type* in the table below shows the first element in the sequence desired.

Argument	Interpretation
1	**Arabic (default for all levels)**
A	**Upper-case alphabetic**
a	**Lower-case alphabetic**
I	**Upper-case roman**
i	**Lower-case roman**

Figure 1: Arguments to the **.AL** Macro

If you specify a *text-indent* argument, the formatter uses it as the number of spaces from the current indent to the text of the list items. This value overrides that in **Li** for the list where you use the argument.

If you give a third argument, a blank line will not separate items in the list. However, a blank line will occur before the first item.

3.2.3. Marked Lists (.ML)

The **.ML** macro expects you to specify an arbitrary *mark* that may consist of one or more characters: **.ML** *mark* [*text-indent* [1]]. Text is indented *text-indent* spaces if the second argument is not null; otherwise, the text is indented one more space than the width of *mark*. If the third argument is specified, no blank lines will separate items in the list.

Do not put ordinary (paddable) spaces into the *mark* because the alignment of items is lost if you justify the right margin. Here's a file containing a marked list before formatting:

```
.ML $
.LI
Sales are up.
.LI
Profits are up.
.LE
```

Here's that same list after formatting:

```
                              - 1 -


$ Sales are up.

$ Profits are up.
```

3.2.4. Variable-Item Lists (.VL)

Another version of the marked list is the "variable-item" list that you call with the **.VL** macro: **.VL** *text-indent* [*mark-indent* [1]]. When you begin a list with a **.VL** macro, there is effectively no current *mark*; you provide each **.LI** its own mark. This form is typically used to display definitions of terms or phrases.

```
.tr ~
.VL 1.0i
.LI requests
are the most elementary text-formatting command available with
the Documenter's Toolkit Software.
.LI macros
are collections of simple formatting commands called by a
single name.
.LI demonstration_of_a_long_mark:
This item shows the effect of a long mark; one space
separates the mark from the beginning of the text.
.LI ~
This item effectively has no mark because the tilde
is translated into a space.
.LE
```

when formatted yields

- 1 -

No hyphenation. Automatic hyphenation is turned off. Words containing hyphens (for example, mother-in-law) may still be split across lines.

Hyphenate. Automatic hyphenation is turned on.

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.

As with the other list types, *text-indent* provides the distance from current indent to beginning of the text. *Mark indent* produces the number of spaces from current indent to beginning of the *mark*, and it defaults to 0 if omitted or null. If you specify a third argument, no blank lines will separate items in the list. Again, do not put ordinary (paddable) spaces into the *mark* because the alignment of items is lost if you justify the right margin.

3.2.5. Bullet, Dash, and Reference Lists (.BL, .DL, .RL)

To initialize any of these lists, type: **.BL or .DL or .RL** [*text-indent* [1]]

A bullet (•) followed by one space marks each list item. As always, if you specify a *text-indent* argument, it overrides the default indentation. In the default case, the text of a bullet list lines up with the first line of indented paragraphs (set with the number register **Pi** {3.1}).

With each of these list types, no blank lines will separate items in the list if you specify a second argument. A dash (—) followed by one space marks each list item of a dash-list. Here's an example of input:

```
.DL
.LI
Prepare documents and tables
.LI
Develop new macro commands
.LE
```

```
                    - 1 -


- Prepare documents and tables

- Develop new macro commands
```

An **.RL** macro call begins an automatically numbered list that encloses the numbers in square brackets ([]).

Here's the input:

```
.RL 8 1
.LI
Using the Documenter's Tool Kit on the DG/UX System
.LI
Documenter's Tool Kit Technical Summary for the DG/UX System
.LE
```

The output is as follows:

```
                    - 1 -


[1]  Using the Documenter's Tool Kit on the DG/UX System
[2]  Documenter's Tool Kit Technical Summary for
     the DG/UX System
```

3.2.6. Nested Lists

Lists may be nested up to six levels. Here is an example of nested lists:

```
.AL
.LI
Develop methods for producing
documentation
.LI
Perform duties resulting from the development of these methods.
For example,
.BL
.LI
Use text processing to:
.DL
.LI
Prepare documents and tables
.LI
Develop new macro commands
.LE
.LI
Serve as a point of contact with printers and
distributors.
.LE
.LI
If the job holder's interests and writing skills match the
needs of the Technical Writing Staff,
write documents.
.LE
```

Here's how that same list looks after it has been formatted.

```
                    - 1 -


1.  Develop methods for producing documentation

2.  Perform duties resulting from the development of these
    methods.  For example:

      ŧ Use text processing to:

          - Prepare documents and tables

          - Develop new macro commands

      ŧ Serve as a point of contact with printers and
        distributors.

3.  If the job holder's interests and writing skills
    matched the needs of the Technical Writing Staff,
    there might be an opportunity to write documents.
```

In this example, the first list-initialization macro that occurs is **.AL**. Since you specify no argument for **.AL**, the formatter numbers list items in sequence with Arabic numerals. Before the list-end macro associated with **.AL** occurs, another list-initialization macro appears, **.BL**. Now, a bullet marks each list item. Finally, **.DL** marks list items with dashes. When the **.LE** associated with **.DL** occurs, a bullet marks list items again, since **.BL** was the list-initialization macro active before the dash list. When **.LE** ends the bullet list, list items are numbered until **.LE** occurs again.

Every time a new list-initialization macro occurs, the list status (indentation, marking style, and so on) changes. **.LE** restores the status generated by the immediately preceding list-initialization macro.

3.2.7. List-Begin Macro and Customized Lists

List-initialization macros described above suffice for almost all cases. However, you may obtain more control over the layout of lists by using the basic list-begin macro (**.LB**). The syntax is as follows:

.LB *text-indent mark-indent pad type* [*mark* [*LI-space* [*LB-space*]]].

The other list-initialization macros use **.LB**. Its arguments are as follows:

- The *text-indent* argument that provides the number of spaces that text indents from the current indent. Normally, this value is taken from the **Li** register (for automatic lists) or from the **Pi** register (for bullet and dash lists).
- The combination of *mark-indent* and *pad* arguments determines the placement of the mark. The mark is placed within an area (called *mark area*) that starts *mark-indent* spaces to the right of the current indent and ends where the text begins (that is, ends *text-indent* spaces to the right of the current indent). The *mark-indent* argument is typically 0.
- Within the *mark area*, the mark is left justified if the *pad* argument is 0. If *pad* is a number *n* (greater than 0), then *n* blanks append to the mark; the *mark-indent* value is ignored. The resulting string immediately precedes the text. The *mark* is effectively right justified *pad* spaces immediately to the left of text.
- The arguments *type* and *mark* interact to control the type of marking used. If *type* is 0, simple marking is performed using the mark character(s) found in the *mark* argument. If *type* is greater than 0, automatic numbering or alphabetizing is done; and *mark* is then interpreted as the first item in the sequence to be used for numbering or alphabetizing. That is, it is chosen from the set (1, A, a, I, i) {3.5.2.6}. This is summarized below:

Type	Mark	Result
0	*omitted*	**hanging indent**
0	*string*	*string* is the mark
>0	*omitted*	Arabic numbering
>0	**1, A, a, I, i**	automatic numbering or alphabetic sequencing

Figure 2: Type and Mark for **.LB**

Each non-zero value of *type* from one to six selects a different way of displaying the marks. The following table shows the output appearance for each value of *type*, where *x* is the generated number or letter:

Value	Appearance
1	*x*.
2	*x*)
3	(*x*)
4	[*x*]
5	<*x*>
6	{*x*}

Figure 3: Appearance for Values of **.LB** Type

- Do not put ordinary (paddable) spaces in the mark.
- The *LI-space* argument gives the number of blank lines (each one-half of the current vertical spacing) that should be output by each .LI macro in the list. If omitted, *LI-space* defaults to 1; use the value 0 to obtain compact lists. If *LI-space* is greater than 0, the .LI macro issues a **.ne** request for two lines just before printing the mark.
- The *LB-space* argument is the number of blank lines (each one-half the vertical spacing) to be output by **.LB** itself. If omitted *LB-space* defaults to 0.

There are three combinations of *LI-space* and *LB-space*:

- The normal case is to set *LI-space* to 1 and *LB-space* to 0 yielding one blank line before each item in the list; such a list is usually ended with a **.LE 1** macro to end the list with a blank line.
- For a more compact list, *LI-space* is set to 0, *LB-space* is set to 1, and the **.LE** macro is used at the end of the list. The result is a list with one blank line before and after it.
- If both *LI-space* and *LB-space* are set to 0 and the .LE macro is used to end the list, a list without any blank lines will result.

3.2.8. Defining List Structures

This section is intended only for people who write formatter macros. If you have not written macros, or if you are content with the lists that **mm** provides by default, you may skip this section. To learn about writing macros, check *Using the Documenter's Tool Kit on the DG/UX System* for "The Formatter **nroff**," "The Formatter **troff**," and check this book for the "**nroff/troff** Technical Discussion."

If a large document requires complex list structures, it is useful to be able to define the appearance for each list level only once instead of having to define it at the beginning of each list. For example, you might define a generalized list-initialization macro in such a way that causes each list-nesting level to behave differently from its predecessor or successor. Suppose you want levels 1 through 5 of lists to have the following appearance:

```
                                - 1 -


A.

 [1]

   o

    a)

      +
```

The following code defines a macro (**.aL**) that always begins a new list and determines the type of list according to the current list level. As the example demonstrates, the **mm** list macros use the number register **:g** to determine the current list level; it is 0 if there is no currently active list. Each call to a list-initialization macro increments **:g**, and each **.LE** call decrements it.

```
.de aL
.\"             register g is used as a local temporary
.\"             to save :g before it is changed below
.nr g \n(:g
.if \\ng=0 .AL A\" give me an A.
.if \\ng=1 .LB \n(Li 0 1 4
.if \\ng=2 .BL\" give me a bullet
.if \\ng=3 .LB \n(Li 0 2 2 a
.if \\ng=4 .ML +\" give me a +
..
```

Now, you can use this macro (with **.LI** and **.LE**) instead of **.AL**, **.RL**, **.BL**, **.LB**, and **.ML**. For example, the following input:

```
.aL
.LI
first line.
.aL
.LI
second line.
.LE
.LI
third line.
.LE
```

will yield

```
                    - 1 -


A.  first line.

      [1] second line.

B.  third line.
```

You could take another approach to lists that is similar to the **.H** mechanism. The list-initialization as well as the **.LI** and the **.LE** macros are all included in a single macro. That macro (called **.bL** below) requires an argument to tell it what level of item is required; it adjusts the list level by either beginning a new list or setting the list level back to a previous value, and then it issues a **.LI** macro call to produce the item:

```
.de bL
.\" if argument, that is the level
.ie \\n(.$ .nr g \\$1
.\" if no argument, use current level
.el .nr g \\n(:g
.if (\\ng-\\n(:g)>1 .)D "**ILLEGAL SKIPPING OF LEVEL"
.\"             increasing level by more than 1
.\"             if g > :g, begin new list
.\"             and reset g to current level (.aL changes g)
.if \\ng>\\n(:g \{.aL \\ng-1
.               nr g \\n(:g\}
.\" if :g > g, prune back to correct level
.if \\n(:g>\\ng .LC \\ng
.\"             if :g = g, stay within current list
.LI             \" always, get out an item
..
```

Calling **.bL** without arguments causes it to stay at the current list level. The **.LC** macro (List Clear) removes list descriptions until the level is less than or equal to that of its argument. For example, the **.H** macro includes the "**.LC 0**" call. If you want to resume text at the end of a list, insert the call "**.LC 0**" to clear out the lists completely. The example below illustrates the small amount of input needed by this approach. The input text

```
The quick brown fox jumped over the lazy dog's back.
.bL 1
first line.
.bL 2
second line.
.bL 1
third line.
.bL
fourth line.
.LC 0
fifth line.
```

yields

```
                    - 1 -


The quick brown fox jumped over the lazy dog's back.

  A.  first line.

       [1] second line.

  B.  third line.

  C.  fourth line.
fifth line.
```

3.3. Footnotes (.FS, .FE)

There are two macros that delimit the text of a footnote. The **.FS** (footnote start) macro marks the beginning of the text of a footnote, and the **.FE** (footnote end) macro marks the end:

```
.FS [label]
Footnote text
.FE
```

These macros form a macro pair; you cannot use one macro without the other. Mark the footnoted line of your paper or memo with **"*F"** or with the optional *label*. If you mark your footnoted line with **"*F,"** do not supply a label with **.FS**; footnotes will be numbered automatically. If you use a footnote label, follow **.FS** with the label you have chosen (**.FS** *label*).

The footnote text (enclosed within the macro pair) should immediately follow the word that you footnote in the input text, so that **"*F"** or *label* occurs at the end of a line of input and the next line is the **.FS** macro call. Consider the following examples. The first is input for a numbered footnote:

```
This is the line containing the word\*F
.FS
This is the text of the footnote.
.FE
to be footnoted.
```

Next is a labeled footnote:

```
This is a labeled*
.FS *
The footnote is labeled with an asterisk.
.FE
footnote.
```

Appendix F shows "Sample Footnotes."

Your memo or paper may contain both user-labeled and automatically numbered footnotes. Another **.FS**, a **.DS** (static display {3.6.1}), or a **.DF** (a floating display {3.6.2}) are not permitted between **.FS** and **.FE** macros. If you do not end the text of a footnote with **.FE**, you will probably cause a formatter error. If you require footnotes in the title, the abstract or in a table, note that only labeled footnotes appear properly. Everywhere else, automatically numbered footnotes work fine.

3.3.1. Changing the Format of Footnote Text (.FD)

Use **.FD** to control the hyphenation, right margin justification, and indentation of footnote text, and to control left or right justification of the footnote label when you indent footnote text: **.FD** [*arg* [1]].

The leftmost column of the following table shows the legal arguments to **.FD** [*arg*]. The remaining four columns show the hyphenation, justification and indentation that you obtain with each value of [*arg*]. For additional information concerning the **.ad**, **.na**, **.hy**, and **.nh** requests (which stand for adjust, no adjust, hyphenation, and no hyphenation, respectively), see the "**nroff/troff** Technical Discussion" in this book.

Argument	Hyphenation	Adjust	Text Indent	Label Justification
0*	.nh	.ad	yes	left
1	.hy	.ad	yes	left
2	.nh	.na	yes	left
3	.hy	.na	yes	left
4	.nh	.ad	no	left
5	.hy	.ad	no	left
6	.nh	.na	no	left
7	.hy	.na	no	left
8	.nh	.ad	yes	right
9	.hy	.ad	yes	right
10**	.nh	.na	yes	right
11	.hy	.na	yes	right

* default for the **mmt** command line
** default for **mm** command line

Figure 4: Arguments to the **.FD** Macro

An argument of 11 or greater is equivalent to **.FD 0**. The effect of a null or omitted argument varies according to the command line you use to process your file. If you use the **mm** command, a null or omitted argument is equivalent to **.FD 10**; if you use the **mmt** command, a null or omitted argument is equivalent to **.FD 0**.

If you specify the second argument, automatically numbered footnotes begin again with 1 when a first-level heading is encountered. This is most useful with the "section-page" page numbering scheme. As an example, the input line **.FD "" 1** maintains the default formatting style and causes footnotes to be numbered afresh after each first-level heading in a document.

Hyphenation across pages is inhibited by **mm** except for long footnotes that continue to the following page. If you permit hyphenation, it is possible for the last word on the last line on the current page footnote to be hyphenated. To avoid this, you may specify an even **.FD** argument.

Footnotes are separated from the body of the text by a short line rule. Those that continue to the next page are separated from the body of the text by a full-width rule. In the **troff** formatter, footnotes are set in type two points smaller than the point size used in the body of text.

3.3.2. Spacing Between Footnote Entries (Fs)

Normally, one blank line (a 3-point vertical space) separates footnotes when more than one occurs on a page. To change this spacing, set the **Fs** number register to the desired value. For example, **.nr Fs 2** will cause two blank lines (a 6-point vertical space) to occur between footnotes.

3.4. Page Headers and Footers

A page header (or header) is text that occurs at the top of pages, while a page footer (or footer) occurs at the bottom of pages. By default, the **mm** macro package centers the page number surrounded by dashes at the top of every page (except the first page of a formal memorandum) and does not print a page footer.

Change this default by using a **mm** page header macro or a page footer macro. Usually, you change a header or footer once at the beginning of the document, but you may change the header or footer as many times as you wish. You may specify a line on every page, a line on the even page only, and a line on the odd page only; thus, the header and footer may contain as many as two lines of text: the line printed at the top of every page and the line for the even- or odd-numbered page.

3.4.1. Page Headers (.PH)

Use the **.PH** macro to specify a header for the top of every page: **.PH** [*arg*]. The initial value of [*arg*] for **.PH** is the centered page number surrounded by dashes.

For all header and footer macros (**.PH**, **.EH**, **.OH**, **.PF**, **.EF**, and **.OF**) the argument [*arg*] is of the form:

```
"'left-part'center-part'right-part'"
```

The formatter left justifies the left-part, centers the center-part, and right justifies the right-part of the header or footer argument that you provide. For example,

```
.PH "'John Smith''Technical Writing Staff'"
```

produces

```
John Smith                                    Technical Writing Staff
```

at the top of every page of the document (after you call **.PH**). In the example above, the center part of the header is left unspecified. If it is inconvenient to use apostrophe (') as the delimiter because an apostrophe occurs within one part, you may uniformly replace the apostrophe with any other character. For example,

```
"*Let's put this left*This center*Let's put this right*"
```

3.4.2. Even-Page Header and Odd-Page Header (.EH, .OH)

The **.EH** macro supplies a line to be printed at the top of each even-numbered page immediately following the page header: **.EH** [*arg*]

The **.OH** macro is the same as the .EH except that it applies to odd-numbered pages: **.OH** [*arg*]. The initial value of [*arg*] for both **.EH** and **.OH** is a blank line.

3.4.3. Page Footer (.PF)

The **.PF** macro specifies the line that is to appear at the bottom of every page: **.PF** [*arg*]. The initial value of the page footer is a blank line.

3.4.4. Even-Page Footer, Odd-Page Footer, and First-Page Footer (.EF, .OF)

The **.EF** macro supplies a line to be printed at the bottom of each even-numbered page immediately preceding the page footer: **.EF** [*arg*]. The **.OF** macro supplies a line to be printed at the bottom of each odd-numbered page immediately preceding the footer: **.OF** [*arg*]. The initial value of these footers is a blank line.

3.4.5. Headers and Footers for the Memorandum and Released Paper Style

In a memorandum or a released-paper style document, the page header on the first page is automatically suppressed provided a break does not occur before the **.MT** macro is called. Macros and text in the following categories do not cause a break and are permitted before the memorandum types (**.MT**) macro:

- Memorandum and released-paper style document macros (**.TL, .AU, .AT, .TM, .AS, .AE, .OK, .ND, .AF, .NS,** and **.NE**)
- Page headers and footers macros (**.PH, .EH, .OH, .PF, .EF,** and **.OF**)
- The **.nr** and **.ds** requests.

3.4.6. Strings and Registers in Header and Footer Macros

String and register names may be placed in arguments to header and footer macros. If the value of the string or register is to be computed when the respective header or footer is printed, invocation must be escaped by four backslashes. This is because string or register invocation is processed three times:

1. As the argument to the header or footer macro
2. In a formatting request within the header or footer macro
3. In a **.tl** request during header or footer processing.

In paragraphs, you only need one backslash (for example, **\nP**). In a page header, you need four; in a static display, you need two, and so on.

For example, the **mm** page number register **P** must be escaped with four backslashes to specify a header in which the page number is to be printed at the right margin, for example: **.PH "'''Page \\\\nP'"** creates a right-justified header containing the word "Page" followed by the page number. Similarly, to specify a footer with the "section-page" style, you specify **.PF "'''– \\\\n(H1–\\\\nP –'"**

If you make the string **a]** contain the current section heading that is to be printed at the bottom of each page, the **.PF** macro call is **.PF "''*(a]''"**.

If you use only one or two backslashes, the footer would print a constant value for **a]**, namely, its value when **.PF** appeared in the input text.

3.4.7. Top and Bottom (Vertical) Margins (.VM)

The .VM (vertical margin) macro allows you to specify additional space at the top and bottom of the page: **.VM** [*top*] [*bottom*]. This space precedes the page header and follows the page footer. A null *top* or *bottom* argument or an argument of 0 puts no additional space before the header or after the footer.

top and *bottom* are two unscaled arguments that are treated as **v**'s (default vertical line spaces: see the "**nroff/troff** Technical Discussion"). For example, **.VM 10 15** adds 10 blank lines to the default top of page margin and 15 blank lines to the default bottom of page margin. Both arguments must be positive (you may decrease default spacing at the top of the page by redefining **.TP** {3.4.9}).

3.4.8. Private Documents (Pv)

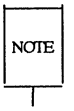
The word "PRIVATE" may be printed, centered, and underlined on the second line of a document (preceding the page header). This is done by setting the **Pv** register value: **.nr Pv** *value*. Possible values are as follows:

Value	Meaning
0	**do not print PRIVATE (default)**
1	**PRIVATE on first page only**
2	**PRIVATE on all pages**

Figure 5: Values for the **Pv** Number Register

If *value* is 2, the user definable **.TP** macro may not be used because **mm** uses the **.TP** macro to print "PRIVATE" on all pages except the first page of a memorandum on which **.TP** is not invoked.

3.4.9. Generalized Top-of-Page Processing



This section is intended only for people who write formatter macros. If you have not written macros, or if you are content the way that **mm** handles top-of-page processing by default, you may skip this section.

During header processing, **mm** invokes two user-definable macros:

- The **.TP** (top of page) macro is invoked in the environment (refer to **.ev** request) of the header.
- The **.PX** is a page header user-exit macro that is invoked (without arguments) when the normal environment has been restored and with the "no-space" mode already in effect.

The effective initial definition of **.TP** (after the first page of a document) is

```
.de TP
.sp 3
.tl \\*(}t
.if e 'tl \\*(}e
.if o 'tl \\*(}o
.sp 2
..
```

The string **}t** contains the header, the string **}e** contains the even-page header, and the string **}o** contains the odd-page header as you define them with the **.PH, .EH,** and **.OH** macros, respectively. To obtain more specialized page titles, you may redefine the **.TP** macro {3.5}. Formatting done within the **.TP** macro is processed in an environment different from that of the body. For example, to obtain a page header that includes three centered lines of data (document number, issue date, and revision date) you could define the **.TP** as follows:

```
.de TP
.sp
.ce 3
777-888-999
Iss. 2, AUG 1985
Rev. 7, SEP 1985
.sp
..
```

Use **.PX** as a user-defined macro to specify text that you want to appear at the top of each page after the normal header.

```
.de PX
.ce
RESTRICTED INFORMATION: FOR YOUR EYES ONLY
..
```

3.5. Section Headings (.H)

mm provides two types of section headings: numbered and unnumbered. To create a numbered section heading, type

.H *level* [*heading-text* [*heading-suffix*]]
Text

The *level* argument provides the numbered heading level. There are seven heading levels; level 1 is the highest, level 7 is the lowest. The *heading-text* argument is the text of the heading. For example,

```
.H 1 "FIRST-LEVEL HEADING"
.H 2 "Second-level heading"
.H 3 "Third-level heading"
.H 1 "ANOTHER FIRST-LEVEL HEADING"
.H 2 "Another second-level heading"
.H 3 "Another third-level heading"
.H 4 "Fourth-level heading"
.H 3 "Still another third-level heading"
.H 5 "Fifth-level heading"
```

produces output like this:

```
1.  FIRST-LEVEL HEADING

1.1  Second-level heading

1.1.1  Third-level heading


2.  ANOTHER FIRST-LEVEL HEADING

2.1  Another second-level heading

2.1.1  Another third-level heading

2.1.1.1  Fourth-level heading

2.1.2  Still another third-level heading

2.1.2.0.1  Fifth-level heading
```

Enclose the argument in double quotes if the heading contains more than one word or contains spaces. One word of heading-text does not require quotes.

In the example above, using a fifth-level heading immediately after a third-level heading makes the value of level 4 become zero. Unless you conform to the hierarchy of headings (using a second-level heading after a first-level heading, and so on), you might obtain results that you do not want.

The *heading-suffix* argument may be used for footnote marks that should not appear with heading text in the table of contents {5.4}. For example,

```
.H 1 "THE UNIX OPERATING SYSTEM" *
.FS *
Trademark of AT&T Bell Laboratories.
.FE
```

Do not use ***F** as the heading suffix. If you do, a number does not appear in the heading as you expect (the string ***F** will) and the footnote numbering goes awry.

There is no need for a **.P** macro {3.1} immediately after **.H** (or **.HU**, see below) because the **.H** macro performs the spacing and indentation functions of the **.P** macro. If you do use **.P** after **.H**, **mm** ignores it. However, it is good practice to start every paragraph of a document with a **.P** macro. Later, if you take headings out of your file, paragraphs remain intact.

The effect of **.H** on line spacing and the font of the heading-text varies according to the *level* argument. Here is the default effect of each *level*.

.H 1 *heading-text*

Produces an underlined (italicized) font heading followed by a single blank line. The text after the *heading-text* begins on a new line and indents according to the current paragraph type.

.H *n heading-text*

Produces an underlined (italicized) heading followed by two spaces ($3 \leq n \leq 7$). The following text begins on the same line, that is, these are run-in headings.

Appropriate numbering and spacing occur even if you omit the *heading-text* argument from a **.H** macro call.

3.5.1. Unnumbered Section Headings (.HU)

To produce an unnumbered heading, type **.HU** *heading-text*. The **.HU** macro is a special case of **.H**; it acts the same way as **.H** except that no heading mark is printed. To preserve the hierarchical structure of headings when you intermix **.H** and **.HU** calls, **.HU** produces headings at level 2 by default. You may change this default value by changing the value of a the number register **Hu**. Whatever value you give **Hu** becomes the heading level for **.HU**. Thus, in the normal case, the only difference between

```
.HU "An unnumbered heading"
```

and

```
.H 2 "A second-level heading"
```

is that the latter prints the heading mark:

```
An unnumbered heading

2.2  A second-level heading
```

By default, both macros have the effect of incrementing the numbering counter for level 2 and resetting to zero the counters for levels 3 through 7. For example,

```
1.  This is a first-level heading

1.1  A second-level heading

1.1.1  A third-level heading

An unnumbered heading

1.2.1  A third-level heading (note that level 2 has incremented)
```

3.5.2. Altering the Appearance of Section Headings

You can change the appearance of headings easily by setting certain registers and strings at the beginning of the document input text file. This permits quick alteration of a document's style because this style-control information is concentrated in a few lines rather than being distributed throughout the document.

3.5.2.1. Prespacing and Page Ejection

A first-level heading, produced by **.H 1**, normally has two blank lines (one vertical space) preceding it. One blank line (one-half vertical space) precedes all other headings. You may force every first-level heading to the top of a new page by inserting **.nr Ej 1** at the beginning of the document input text file. Make long documents more manageable by starting each section on a new page. Setting the **Ej** register to a higher value causes the same effect for headings up to that level, that is, a page eject occurs if the heading level is less than or equal to the **Ej** value.

3.5.2.2. Spacing after Section Headings

Three registers control the appearance of text immediately following a **.H** call. The registers are **Hb** (heading break level), **Hs** (heading space level), and **Hi** (post-heading indent).

- If the heading level is less than or equal to **Hb**, a line break {1.4.1} occurs after the heading.
- If the heading level is less than or equal to **Hs**, **mm** inserts a blank line (one-half vertical space) after the heading.
- If a heading level is greater than **Hb** and also greater than **Hs**, then the heading (if any) is immediately followed by text on the same line.

These registers permit you to separate headings from the text in a consistent way throughout a document and allow you to alter easily white space and heading emphasis. The default value for **Hb** and **Hs** is 2.

For any stand-alone heading (a heading on a line by itself) the **Hi** number register controls alignment of the next line of output.

- If **Hi** is 0, text is left-justified.
- If **Hi** is 1 (the default value), **mm** indents text according to the paragraph type as specified by the **Pt** register {3.1.1}.
- If **Hi** is 2, **mm** indents text to line up with the first word of the heading itself so that the heading number stands out more clearly.

To cause a blank line (one-half vertical space) to appear after the first three heading levels, to have no run-in headings, and to force the text following all headings to be left-justified (regardless of the value of **Pt**), you should put the following line in the parameter setting segment:

```
.nr Hs 3
.nr Hb 7
.nr Hi 0
```

3.5.2.3. Centered Section Headings (Hc)

Use the **Hc** register to obtain centered headings. A heading is centered if its level argument is less than or equal to **Hc** and if it is also a stand-alone heading {3.5.2.2}. For example,

```
.nr Hc 1
.H 1 DG/UX
.H 2 "Application Packages"
.H 2 Languages
```

produces

1. DG/UX

1.1 Application Packages

1.2 Languages

The **Hc** register is initially set to 0 (no centered headings).

3.5.2.4. Bold, Italic, and Underlined Headings

3.5.2.4.1. Control by Level.

Any heading that is underlined by the **nroff** formatter is italicized by the **troff** formatter. The string **HF** (heading font) contains seven codes that specify fonts for heading levels 1 through 7. Legal codes, code interpretations, and defaults for **HF** codes are shown below:

Formatter	HF			Default HF
	1	2	3	
nroff	no underline	underline	bold	2 2 2 2 2 2 2
troff	roman	italic	bold	2 2 2 2 2 2 2

Figure 6: The **HF** String

Thus, all levels are underlined by the **nroff** formatter and italicized by the **troff** formatter. You may reset **HF** as desired. Any value omitted from the right end of the list is assumed to be a 1. The following request would result in five bold levels and two underlined (italic) levels:

```
.ds HF 3 3 3 3 3 2 2
```

3.5.2.4.2. nroff Underlining Style.

The **nroff** formatter underlines in either of two styles:

- The normal style (**.ul** request) is to underline only letters and digits.
- The continuous style (**.cu** request) underlines all characters including spaces.

By default, **mm** attempts to use the continuous style on any heading that is to be underlined and is short enough to fit on a single line. If a heading is to be underlined but is longer than a single line, the heading is underlined in the normal style.

All underlining of headings can be forced to the normal style by using the **−rU1** flag when invoking the **nroff** formatter {8.4}.

3.5.2.5. Section Heading Point Sizes (HP)

You may specify the desired point size for each heading level with the *HP* string (for use with the **troff** formatter only).

```
.ds HP [ps1] [ps2] [ps3] [ps4] [ps5] [ps6] [ps7]
```

By default, **mm** prints the text of headings (**.H** and **.HU**) in the same point size as the body except that bold stand-alone headings are printed in a size one point smaller than the body. You can specify the string **HP**, which is similar to the string **HF**, to contain up to seven values, corresponding to the seven levels of headings. For example,

```
.ds HP 12 12 10 10 10 10 10
```

specifies that the first and second level headings are to be printed in 12-point type with the remainder printed in 10-point. Specified values may also be relative point-size changes, for example,

```
.ds HP +2 +2 −1 −1
```

If you specify absolute point sizes, then absolute sizes are used regardless of the point size of the body of the document. If relative point sizes are specified, then point sizes for headings are relative to the point size of the body even if the latter is changed.

Null or zero values imply that the default size is used for the corresponding heading level. Only the point size of the headings is affected. Specifying a large point size without providing increased vertical spacing (via **.HX** and/or **.HZ** {3.5.4}) may cause overprinting.

3.5.2.6. Marking Styles Numerals and Concatenation (.HM)

The registers named **H1** through **H7** are used as counters for the seven levels of headings. Register values are normally printed using Arabic numerals. The **.HM** macro (heading mark style) allows this choice to be overridden thus providing "outline" and other document styles:

.HM [*arg1*] . . . [*arg7*]

This macro can have up to seven arguments; each argument is a string indicating the type of marking to be used. Legal arguments and their meanings are

Argument	Meaning
1	Arabic (default for all levels)
0001	Arabic with enough leading zeroes to get the specified number of digits
A	Upper-case alphabetic
a	Lower-case alphabetic
I	Upper-case roman
i	Lower-case roman
omitted	Interpreted as 1 (Arabic)
illegal	No effect

By default, the complete heading mark for a given level is built by concatenating the mark for that level to the right of all marks for all levels of higher value. To inhibit the concatenation of heading level marks, that is, to obtain just the current level mark followed by a period, the heading mark type register (**Ht**) is set to 1. For example, a commonly used "outline" style is obtained by:

```
.HM I A 1 a i
.nr Ht 1
```

3.5.3. Headings and Table of Contents (Cl)

Automatically collect the text of headings and their corresponding page numbers for a table of contents by doing the following:

- Specify in the contents level register, **Cl**, what level headings you want to save
- Call the **.TC** macro {5.4} at the end of the document.

mm saves any heading whose level is less than or equal to the value of the **Cl** register and later displays it in the table of contents. The first two levels of headings are saved if you use **.TC** without putting a value in **Cl** (that is, **Cl** by default contains the value 2).

Because of the way headings are saved, it is possible to exceed the formatter's storage capacity, particularly when saving many levels of many headings, while also processing displays {3.6} and footnotes {3.3}. If this happens, the "Out of temp file space" formatter error message appears (see Appendix D); the only remedy is to save fewer levels and/or to have fewer words in the heading text.

3.5.4. Section Headings and User Exit Macros

The **.HX**, **.HY**, and **.HZ** macros are the means by which you obtain a final level of control over the section heading mechanism:

.HX *dlevel rlevel heading-text*
.HY *dlevel rlevel heading-text*
.HZ *dlevel rlevel heading-text*

These macros are not defined by **mm**; they are intended to be defined by you. The **.H** macro call invokes **.HX** shortly before it prints the heading text; it calls **.HZ** as its last action. After **.HX** is invoked, the size of the heading is calculated. This processing causes certain features that may have been included in **.HX**, such as **.ti** for temporary indent, to be lost. After the size calculation, **.HY** is called so that you may redefine these features. All default actions occur if these macros are not defined. If **.HX**, **.HY**, or **.HZ** are defined by you, user-supplied definition is interpreted at the appropriate point. These macros can therefore influence handling of all headings because the **.HU** macro is actually a special case of the **.H** macro.

If you originally invoked the **.H** macro, then the derived level argument (*dlevel*) and the real level argument (*rlevel*) are both equal to the level given in the **.H** invocation. If you originally invoked the **.HU** macro {3.5.1}, *dlevel* is equal to the contents of register **Hu**, and *rlevel* is 0. In both cases, *heading-text* is the text of the original invocation.

By the time **.H** calls **.HX**, it has already incremented the heading counter of the specified level {3.5.2.6}, produced blank lines (vertical spaces) to precede the heading {3.5.2.1}, and accumulated the "heading mark," that is, the string of digits, letters, and periods needed for a numbered heading.

When **.H** calls **.HX**, you may reference all **mm** registers and strings, as well as the following:

string **}0** If you make *rlevel* non-zero, this string contains the "heading mark." Two unpaddable spaces (to separate the *mark* from the *heading*) have been appended to this string.

If *rlevel* is 0, this string is null. If string **}0** is null, you omit the heading mark in the table of contents produced with the **.TC** macro.

register **;0** This register shows the type of spacing that is to follow the heading {3.5.2.2}.

A value of 0 means that the heading is run-in. A value of 1 means a break (but no blank line) is to follow the heading. A value of 2 means that a blank line (one-half vertical space) is to follow the heading.

string }2 If "register **;0**" is 0, this string contains two unpaddable spaces that will be used to separate the (run-in) heading from the following text.

If "register **;0**" is non-zero, this string is null.

register **;3** This register contains an adjustment factor for a **.ne** request issued before the heading is actually printed. On entry to **.HX**, it has the value 3 if *dlevel* equals 1, and 1 otherwise. The **.ne** request is for the following number of lines: the contents of the "register **;0**" taken as blank lines (halves of vertical space) plus the contents of "register **;3**" as blank lines (halves of vertical space) plus the number of lines of the heading.

You may alter the values of }0, }2, and ;3 within **.HX**. The following are examples of actions that might be performed by defining **.HX** to include the lines shown:

- Change first-level heading mark from format *n.* to *n.*0:
  `.if \\$1=1 .ds }0 \\n(H1.0\<sp>\<sp>`
  (where <sp> stands for a space)

- Separate run-in heading from the text with a period and two unpaddable spaces:
  ```
  .if \\n(;0=0 .ds }2 .\<sp>\<sp>
  ```
- Ensure that at least 15 lines are left on the page before printing a first-level heading:
  ```
  .if \\$1=1 .nr ;3 (15-\\n(;0
  ```
- Add three additional blank lines before each first-level heading:
  ```
  .if \\$1=1 .sp 3
  ```
- Indent level 3 run-in headings by five spaces:
  ```
  .if \\$1=3 .ti 5n
  ```

If temporary strings or macros are used within **.HX**, their names should be chosen with care {6.10.1}.

When **.H** calls the **.HY** macro after the **.ne** is issued, certain features requested in **.HX** must be repeated. For example,

```
.de HY
.if \\$1=3 .ti 5n
..
```

The **.HZ** macro is called at the end of **.H** to control actions after the heading is produced. In a large document, sections may correspond to chapters of a book, and you may want to change a page header or footer, for example:

```
.de HZ
.if \\$1=1 .PF "Section \\$3"
..
```

3.6. Displays

Displays are blocks of text that you want kept together, not split across pages. **mm** provides two styles of displays: static and floating.

A static display appears in the same relative position in the output text as it does in the input text. This may result in extra white space at the bottom of the page if the display is too big to fit there.

A floating display "floats" through the input text to the top of the next page if there is not enough room for it on the current page. Input text that follows a floating display may precede it in the output text.

By default, a display is processed with line-filling turned off, with single-spacing, and not indented from the exiting margins. Do not nest displays and footnotes, in any combination. Do not put headings within displays or footnotes.

3.6.1. Static Displays (.DS, .DE)

A static display is delimited by the **.DS** and **.DE** macro pair.

```
.DS [format [fill [rindent] ] ]
Text
.DE
```

With no arguments, **.DS** accepts lines of text exactly as typed (line-filling off) and will not indent lines from the prevailing left margin or from the right margin.

The *format* argument is an integer or letter you use to control the indentation and centering of displays. The *fill* argument is an integer or letter. These arguments can have the following meanings:

Format	Meaning
""	**no indent**
0 or L	**no indent**
1 or I	**indent by standard amount**
2 or C	**center each line**
3 or CB	**center as a block**
none	**no indent**

Fill	Meaning
""	**line-filling off**
0 or N	**line-filling off**
1 or F	**line-filling on**
none	**line-filling off**

Figure 7: Arguments to the **.DS** Macro

The *rindent* argument is the number of characters that the line length should be decreased, that is, an indentation from the right margin.

The default static display indentation wth **.DS 1** or **.DS I** is five spaces, but you can change it by changing the value in the number register **Si**. By default, then, text of an indented display aligns with the first line of indented paragraphs, unless you also change the value contained in **Pi** {3.1}. These two number registers are independent of one another.

The display *format* argument value **3** (or **CB**) centers (horizontally) the entire display as a block (as opposed to **.DS 2** and **.DF 2** that center each line individually). All collected lines are left justified, and the display is centered based on width of the longest line. By default, a blank line is placed before and after static and floating displays. You can prevent this by setting the number register **Ds** to **0**.

The following example shows usage of all three arguments for static displays. The input

```
.DS I F 5
We the people of the United States,
in order to form a more perfect union,
establish justice, ensure domestic tranquillity,
provide for the common defense,
and secure the blessings of liberty to
ourselves and our posterity,
do ordain and establish this Constitution to the
United States of America.
.DE
```

produces

```
     We the people of the United States, in order to form a more perfect
union, establish justice, ensure domestic tranquillity, provide for the common
defense, and secure the blessings of liberty to ourselves and our posterity, do
ordain and establish this Constitution to the United States of America.
```

This block of text is indented five ems from the current left margin, filled, and indented five spaces from the right margin.

3.6.2. Floating Displays (.DF, .DE)

Delimit a floating display with the macro pair **.DF** and **.DE**.

```
.DF [format [fill [rindent] ] ]
Text
.DE
```

Arguments to **.DF** have the same meanings as they do to **.DS** except when they concern *format*. With floating displays, the formatter calculates indentation and centering with respect to the initial left margin because the prevailing indent may change between the time when the formatter first reads the floating display and when the display is printed. One blank line occurs before and after a floating display.

When the formatter encounters a floating display, it processes and places the display onto a queue waiting to be output. The formatter removes displays from the queue and prints them in the order entered, which is the order they appeared in the input file. If a new floating display is encountered and the queue of displays is empty, the new display is a candidate for immediate output on the current page.

As long as the display queue contains one or more displays, the formatter automatically enters new displays there, rather than putting them out. When the formatter puts out a display, it also removes it from the queue.

When the formatter reaches the end of a section (using section-page numbering) or the end of a document, it automatically removes all displays from the queue, putting them out. This occurs before the formatter processes an **.SG** macro {5.1}.

A display will fit on the current page if there is enough room to contain the entire display or if the display is longer than one page in length and less than half of the current page has been used.

You may exercise precise control over the positioning of floating displays on output with two number registers, **De** and **Df** (see below). Immediate output of the display queue is governed by size of display and the setting of the **Df** register code. The **De** register code controls whether text will appear on the current page after a floating display has been produced.

The **De** and **Df** number register code settings and actions are as follows:

De register:

Code	Action
0	No special action occurs (also the default condition).
1	A page eject always follows the output of each floating display, so only one floating display appears on a page and no text follows it.

For any other code, the action performed is the same as for code 1.

Df register:

Code Action

0 Floating displays are not output until end of section (when section-page numbering) or end of document.

1 Output new floating display on current page if there is space; otherwise, hold it until end of section or document.

2 Output exactly one floating display from queue to the top of a new page or column (when in 2-column mode).

3 Output one floating display on current page if there is space; otherwise, output to the top of a new page or column.

4 Output as many displays as will fit (at least one) starting at the top of a new page or column. If **De** is set to 1, each display is followed by a page eject, causing a new top of page to be reached where at least one more display is output.

5 Output a new floating display on the current page if there is room (default condition). Output as many displays (but at least one) as will fit on the page starting at the top of a new page or column. If **De** is set to 1, each display is followed by a page eject causing a new top of page to be reached where at least one more display is output.

For any code greater than 5, the action performed is the same as for code 5.

You may also use the **.WC** macro {6.7} to control handling of displays in double-column mode and to control the break in text before floating displays.

3.7. Tables (using tbl)

```
.TS [H]
global options;
format section.
title lines
[.TH [N]]
Data
.TE
```

The macro pair **.TS** (table start) and **.TE** (table end) delimits text to be examined by **tbl** and sets proper spacing around the table. The display function (**.DS** and **.DE**) and the **tbl** delimiting function are independent. To keep together blocks that contain any mixture of tables, equations, filled text, unfilled text, and caption lines, enclose

the **.TS/.TE** block within a display (**.DS/.DE**) if the table is less than a page long. You may enclose floating tables inside floating displays (**.DF/.DE**).

mm formats headings for tables that extend over several pages. If a table heading is needed for each page of a multi-page table, the **H** argument should be specified to the **.TS** macro as above. Following the options and format information, table title is typed on as many lines as required and is followed by the **.TH** macro. The **.TH** macro must occur when **.TS H** is used for a multi-page table. This is not a feature of **tbl** but of the definitions provided by the **mm** macro package.

The **.TH** (table header) macro may take as an argument the letter **N**. This argument causes the table header to be printed only if it is the first table header on the page. Use this option when it is necessary to build long tables from smaller **.TS H/.TE** segments. For example,

```
.TS H
Global options;
Format section.
Title lines
.TH
Data
.TE
.TS H
Global options;
Format section.
Title lines
.TH N
Data
.TE
```

causes the table heading to appear at the top of the first table segment and no heading to appear at the top of the second segment when both appear on the same page. However, the heading still appears at the top of each page that the table continues onto. Use this feature when a single table must be broken into segments because of table complexity (for example, too many blocks of filled text). If each segment had its own **.TS H/.TH** sequence, it would have its own header. However, if each table segment after the first uses **.TS H/.TH N**, the table header will appear only at the beginning of the table and the top of each new page or column that the table continues onto.

For the **nroff** formatter, you may use the **−e** option [**−E** for **mm** {8.1}] for terminals that are capable of finer printing resolution. This causes better alignment of features such as the lines forming the corner of a box. The **−e** option is not effective with **col**.

3.8. Equations (using eqn)

```
.DS
.EQ [label]
Equation(s) input
.EN
.DE
```

The programs **neqn** and **eqn** expect to use the **.EQ** (equation start) and **.EN** (equation end) macros as delimiters in the same way that **tbl** uses **.TS** and **.TE**; however, **.EQ** and **.EN** must occur inside a **.DS/.DE** pair. There is an exception to this rule – if **.EQ** and **.EN** are used to specify only the delimiters for in-line equations or to specify **eqn/neqn** defines, the **.DS** and **.DE** macros must not be used; otherwise, extra blank lines will appear in the output.

The **.EQ** macro takes an argument that will be used as a label for the equation. By default, the label will appear at the right margin in the "vertical center" of the general equation. The **Eq** register may be set to 1 to change labeling to the left margin.

The equation will be centered for centered displays; otherwise, the equation will be adjusted to the opposite margin from the label.

3.9. Figure, Table, Equation, and Exhibit Titles (.FG, .TB, .EC, .EX)

You may use the **.FG** (figure title), **.TB** (table title), **.EC** (equation caption), and **.EX** (exhibit caption) macros inside **.DS/.DE** pairs to number figures, tables, and equations automatically, and give them titles.

```
.FG [title [override [flag] ] ]
.TB [title [override [flag] ] ]
.EC [title [override [flag] ] ]
.EX [title [override [flag] ] ]
```

These macros use registers **Fg**, **Tb**, **Ec**, and **Ex**, respectively (see section 8.4 on **–rN5** to reset counters in sections). For example,

```
.FG "This is a Figure Title"
```

yields

Figure 1. This is a Figure Title

The **.TB** macro replaces "Figure" with "TABLE," the **.EC** macro replaces "Figure" with "Equation," and the **.EX** macro replaces "Figure" with "Exhibit." The output title is centered if it can fit on a single line; otherwise, all lines but the first are indented to line up with the first character of the title. Change the format of the numbers using the **.af** request of the formatter. By setting the **Of** register to 1, you may change the format of the caption from

Figure 1. Title

to

Figure 1 - Title

Use the *override* argument to change normal numbering. If you omit the *flag* argument or use an argument of 0, *override* is used as a prefix to the number; if the *flag* argument is 1, *override* becomes a suffix; and if the *flag* argument is 2, *override* replaces the number. If **-rN5** {8.4} is given, "section-figure" numbering is set automatically and user-specified *override* argument is ignored.

As a matter of formatting style, you might want to place table headings above the text of tables, and put figure, equation, and exhibit titles below corresponding figures and equations.

You obtain a List of Figures, List of Tables, List of Exhibits, and List of Equations after **mm** prints the Table of Contents if the number registers **Lf, Lt, Lx** and **Le** (respectively) are set to 1. By default, all but **Le** are set to 1 by default. You can change the titles of these lists by redefining the following strings, which are presented here with their default values:

```
.ds Lf LIST OF FIGURES
.ds Lt LIST OF TABLES
.ds Lx LIST OF EXHIBITS
.ds Le LIST OF EQUATIONS
```

4. Formatting the Beginning of a Document

Two specific styles of documents are available with **mm**: formal memorandum style, which includes formats for memoranda, released papers and external letters, and business letter style.

4.1. Formal Memorandum Style

The **mm** formal memorandum style allows three document types: the memorandum, the released paper and the external letter. Commonly, people who write formal memoranda put certain information at the beginning of the document (the date, title, case numbers, authors, and so on) or at the end of the document (the signature line and a list of the document's recipients), and put it nowhere else. You specify these beginning and end items the same way for each formal memorandum type. Their formatted appearance depends on which type you choose with the **.MT** macro. (See "Appendix G" for an example of a formal memorandum.)

4.1.1. Choosing a Formal Memorandum Type (.MT)

This is how you use **.MT**:

.MT [*argument* [*addressee*]]

An *argument* specifies a particular formal memorandum type. Legal values for the *argument* are as follows:

Argument	Type	Value
	memorandum	no memorandum type printed
0	memorandum	no memorandum type printed
none	memorandum	MEMORANDUM FOR FILE
1	memorandum	MEMORANDUM FOR FILE
2	memorandum	PROGRAMMER'S NOTES
3	memorandum	ENGINEER'S NOTES
4	released-paper	released-paper style
5	external-letter	external-letter style
"*string*"	memorandum	*string* (enclosed in quotes)

Figure 8: Arguments to the **.MT** Macro

The formal memorandum style produces a standard mast at the top of the first page of your document. The following input lines produce the next mast (and those that follow it). Put these lines immediately after the parameter setting segment of your document:

```
.ND "September 28, 1984"
.TL
Document Production Coordinator
.AU "John Smith" JS XF 5414 6398 7-123
.AF "Business Computer Systems, Inc."
.MT n (where n is a legal argument to .MT)
```

First, consider the memorandum type (here, **.MT**).

Business Computer Systems, Inc.

subject: Document Production Coordinator

date: September 28, 1984

from: John Smith
XF 5414
x6398 7-123

If you give **.MT** any argument other than 4 or 5, you obtain, a few lines after the last line of author information, the value of *value* in the preceding table.

There are two alternatives to the memorandum type. To obtain the released-paper style, use **.MT 4**, which produces a different mast:

```
Document Production Coordinator

John Smith

Business Computer Systems, Inc.
```

With the external-letter style (**.MT 5**), **mm** prints only the title (without the word "subject:") and the date in the opposite left and right corners, respectively, of the top of the first page.

```
Document Production
Coordinator

                                                    September 28, 1984
```

Specify the addressee of a memo, released paper or letter with the second argument to **.MT**. This argument may be any words you choose. Providing the addressee causes the name you've specified and the page number to replace the normal page header (page headers will be discussed below) on the second and succeeding pages of a memo.

```
.MT 1 "Michael Smith"
```

You may not use the addressee argument when the **.MT** type equals 4. If you try, output will cease after the first page.

4.1.2. TM Numbers

If the memorandum is an AT&T technical memorandum, TM numbers are supplied via the **.TM** macro.

.TM [*number*] ...

Up to nine numbers may be specified. For example,

```
.TM 7654321 77777777
```

mm ignores this macro call in the released-paper and external-letter styles {4.4.1}.

4.1.3. Changing the Date (.ND)

By default, the current date appears in the "date" part of a memorandum or in the right corner of an external letter. You may override the current date using the **.ND** macro.

.ND *new date*

4.1.4. Giving the Memorandum a Title (.TL)

The **.TL** macro gives your formatted document a title. To use **.TL**, type:

.TL [*charging-case number(s)* [*filing-case number(s)*]]
Text
.AU *(or another macro)*

AT&T Bell Laboratories uses arguments to the **.TL** macro to specify the *charging-case number(s)* and *filing-case number(s)*.

- The *charging-case number* stands for an account to which a person's time is charged. Enter multiple *charging-case numbers* as "subarguments" by separating each from the previous with a comma and a space, and enclosing the entire

argument within double quotes (see example below).

- The *filing-case number* describes where the memorandum is to be filed. Enter multiple *filing-case numbers* the same way you enter *charging-case numbers* (see example below).

Here is an example of specifying more than one *charging-case number* and *filing-case number*:

```
.TL "12345, 67890" "987654321, 987654322"
Document Production Coordinator
```

Those numbers will appear after the title like this (except for released paper style, when they do not appear at all):

```
Document Production
Coordinator
Charge Case 12345, 67890
File Case 987654321, 987654322
```

The title of the memorandum follows the **.TL** macro. You may use the **.br** request to break the title into several lines

```
.TL 12345
Document Production
.br
Coordinator for the
.br
Technical Writing Staff
```

4.1.5. Specifying the Author (.AU)

Use **.AU** to specify the author of your memo or paper:

```
.AU name [initials [loc [dept [ext [room [arg [arg [arg]]]]]]]]
.TL 12345
Document Production
.br
Coordinator for the
.br
Technical Writing Staff
```

In the "from:" portion of a formatted memorandum, location and department number follows the author's name on one line and room number and extension number follow it on the next line. The "x" for the extension is added automatically:

```
from: John Smith
      XF 5414
      7-123 x6398
      machine_5!jjs
```

For the memorandum type, you type information that describes an author after the **.AU** macro as arguments. This information includes:

- name (for example, John Smith)
- initials (for example, JJS)
- location (XF)
- department (5414)
- telephone extension (6398)
- room (7-123)
- one, two or three additional arguments (for example, machine_5!jjs)

The first six arguments must appear in the order given, that is, the author's name must be typed before the initials, which must be typed before the location, and so on. By default, these arguments are ignored for the released paper style and the external letter style. If you want to leave any of these arguments blank, put a null argument at the appropriate place.

If you want to suppress printing the location, department number, extension number, room number and later arguments, set the number register **Au** to 0; the default value is 1.

If a memorandum has more than one author, use a separate **.AU** macro for each author, for example,

```
.AU "John Smith" JJS XF 5414 6398 7-123 machine_5!jjs
.AU "John Foley" JJF XF 5415 6666 7-321 machine_6!jf
```

produces

```
from: John Smith
      XF 5414
      7-123 x6398
      machine_5!jjs

      John Foley
      XF 5415
      7-321 x6666
      machine_6!jf
```

4.1.6. Specifying the Author's Title (.AT)

Specify the author's title with the **.AT** macro.

.AT *title ...*

.AT must immediately follow **.AU** for the given author. For example,

```
.AU "John Smith" JJS XF 5414 6398 7-123
.AT Supervisor "Technical Writing Staff"
```

produces the following output at the signature block:

```
John Smith
Supervisor
Technical Writing Staff
```

You may give **.AT** up to nine arguments. Each argument will appear in the signature block (at the end of the memorandum, which is discussed below) on a separate line following the signer's name. If you need a long title, surround phrases in double quotes, turning several words into single arguments.

4.1.7. Specifying the Author's Firm (.AF)

Supply the name of your firm with **.AF**.

.AF "*name of the firm*"

Use **.AF** before **.AU** to avoid a formatting error. If you use **.MT 4**, your firm name appears after the author's name. For example,

```
.AF "Business Computer Systems, Inc."
```

puts "Business Computer Systems, Inc." in bold letters in the upper right hand corner of the first page of your memo. If you specify **.MT 4**, "Business Computer Systems, Inc." appears centered and double spaced from the author's name.

If you do not supply a name with **.AF**, "AT&T Bell Laboratories" appears as the name of your firm unless your system administrator edits **strings.mm** {5.5}.

4.1.8. Calling Beginning Formal Memorandum Macros in the Correct Order

If you use the macros described thus far, you must call them in the following order to avoid a formatting error:

```
.ND new date
.TL [charging-case number(s) [filing-case number(s)] ]
Text
.AF "name of the firm"
.AU Name [initials [loc [dept [ext [room [arg [arg [arg]]]]]]]]
.MT [type [addressee] ]

.AF "Business Computer Systems, Inc."
```

The only required macros for a memorandum, released paper or external letter are **.TL**, **.AU**, and **.MT**.

4.2. Other Beginning Macros

4.2.1. Abstract (.AS, .AE)

If a formal memorandum has an abstract, delimit the text of the abstract with the **.AS** (abstract start) and **.AE** (abstract end) macro pair.

```
.AS [arg [indent] ]
Text of abstract
.AE
```

Abstracts are printed on page one of a document and/or on its cover sheet. There are three types of cover sheets:

- Released paper
- Memorandum
- Memorandum for file (also used for engineer's notes, memoranda for record, and so on)

Cover sheets for released papers and memoranda are obtained by invoking the **.CS** macro.

With the released-paper type (argument to the **.MT** macro is 4) and with the memorandum type, if the first argument of **.AS** is

0 – Abstract prints on page 1 and on the cover sheet (if any).

1 – Abstract appears only on the cover sheet (if any).

With the memorandum for file type and in all other documents (other than external letters) if the first argument of **.AS** is:

0 –Abstract appears on page 1 and no cover sheet prints.

2 –Abstract appears only on the cover sheet that will be produced automatically (that is, without invoking the .CS macro).

It is not possible to get either an abstract or a cover sheet with an external letter (first argument of the **.MT** macro is 5).

Notations such as a "Copy to" list are allowed on memorandum for file cover sheets; the **.NS** and **.NE** macros must appear after the **.AS 2** and **.AE** macros. Headings and displays are not permitted within an abstract.

The abstract is printed with ordinary text margins; an indentation to be used for both margins can be specified as the second argument of **.AS**. Values that specify indentation must be unscaled and are treated as "character positions," that is, as the number of ens.

4.2.2. Other Keywords (.OK)

.OK *keyword* [...]

Topical keywords should be specified on a technical memorandum cover sheet. You may specify up to nine such keywords or keyword phrases as arguments to the **.OK** macro; if any keyword contains spaces, you must enclose them in double quotes.

4.2.3. Bottom Block (.BS, .BE)

Specify lines of text to be printed at the bottom of each page after the footnotes (if any) but before the page footer with the bottom block macro pair **.BS/.BE**.

```
.BS
Text
.BE
```

The bottom block should occur before the use of any footnotes {3.3} or macros that define the memorandum style {4.1.4}. Otherwise, an interaction between this macro

pair and another macro that redefines the appearance of the bottom of the page may cause you problems.

Remove the bottom block by specifying an empty block, as shown below:

```
.BS
.BE
```

The bottom block appears on the table of contents, text pages, and the cover sheet for memorandum for file, but it does not appear on the technical memorandum or released-paper cover sheets.

4.3. Proprietary Marking Macro (.PM)

The **.PM** (proprietary marking) macro appends to the page footer a proprietary disclaimer.

.PM [*code*]

The argument is selected from among the following:

Current Arg	Former Arg	Disclaimer Message
PM1	BP,N,P, BPN	AT&T BELL LABORATORIES - PROPRIETARY Use pursuant to G.E.I. 2.2
PM2 CA	*none*	THIS DOCUMENT CONTAINS PROPRIETARY INFORMATION OF AT&T AND IS NOT TO BE DISCLOSED OR USED EXCEPT IN ACCORDANCE WITH APPLICABLE CONTRACTS OR AGREEMENTS.
PM3 CP	*none*	SEE PROPRIETARY NOTICE ON COVER PAGE
PM4	BPP,BR	AT&T BELL LABORATORIES - PROPRIETARY (RESTRICTED) Solely for authorized persons having a need to know pursuant to G.E.I. 2.2
PM5	ILL	THIS DOCUMENT CONTAINS PROPRIETARY INFORMATION OF AT&T BELL LABORATORIES AND IS NOT TO BE DISCLOSED, REPRODUCED, OR PUBLISHED WITHOUT WRITTEN CONSENT. THIS DOCUMENT MUST BE RENDERED ILLEGIBLE WHEN BEING DISCARDED.
PM6	CI-II	CI-II Not for disclosure to AT&T Information Systems. Subject to FCC separation requirements under Computer Inquiry II

Figure 9: Arguments to **.PM** (Proprietary Markings)

Use **.PM** at the beginning of your document, before you use footnotes {3.3} or macros that define the memorandum style {4.1.4}. Otherwise, an interaction between this macro and another that redefines the appearance of the bottom of the page may cause you problems.

These disclaimers are in a form approved for use by the AT&T. Markings are underlined. (They are italic in **troff**.) You may use the CI-II marking with any other message by two separate **.PM** requests. For example,

```
.PM CI-II
.PM N
```

produces a CI-II and NOTICE mark.

These proprietary markings are specified in the define file. System administrators can change the contents of this define file, **strings.mm**, to match your needs. This file is described in the next section. In cases where the disclaimer message for a code argument has been removed the argument issues a currently approved disclaimer message. Since the code argument may produce a different disclaimer message (a shorter or longer message), the page formatting of the document may be affected.

4.4. Define File Information

The define file contains pre-defined strings for the **.MT** and **.PM** macros. Appendix E presents the contents of the file. The file **/usr/lib/macros/strings.mm** contains the define file. Only system administrators may change specific string and font information, since only they have write permissions for the define file.

4.5. Business Letter Style

An alternative to the formal memorandum style is the business letter style, which produces four types of business letters: blocked, semiblocked, full-blocked, and simplified. (See Appendix H for an example of an **mm** business letter.)

4.5.1. Letter-Type Macro (.LT)

The letter-type macro **.LT** formats a letter in one of four business styles:

```
.LT [arg]
```

.LT accepts one (optional) argument. Arguments and corresponding format are as follows:

Argument	Format
none	blocked
BL	blocked
SB	semiblocked
FB	full-blocked
SP	simplified

.LT controls the placement on the page of the output of the subordinate macro **.LO** and the subordinate macro pairs (**.IA** and **.IE**, **.WA** and **.WE**) which differs according to each of the four business letter formats.

Business letter and formal memorandum macros (**.LT** and **.MT**) are mutually exclusive. If you specify both **.LT** and **.MT** specific macros in a single document, **nroff/troff** attempts to process the file according to the first formatting specific macro it encounters. **mm** ignores .MT-specific macros (**.2C**, **.AF**, **.AS**, **.AT**, **.AU**, **.AV**, **.CS**, **.OK**, **.TC**, **.TL**, **.TM**) and .MT-specific command line parameters (**−rAn**, **−rEn**, **−rN4**) if you use them with **.LT**; conversely, if you use **.LT**-specific macros (**.WA**, **.WE**, **.IA**, **.IE**, **.LO**) with **.MT**, **mm** ignores them.

If you use these business letter macros, the macro pairs, **.WA**-**.WE** and **.IA**-**.IE**, and the page formatting macro **.LT** are required, all other business letter macros are optional.

The **.LT** macro arguments control paragraph indentation for each of the four letter types. If you redefine the **Pt** and **Pi** registers, the user specified indentations will override. Specification of the **Pt** and **Pi** registers must occur after specification of the **.LT** macro.

- In the blocked format all lines of text begin at the left margin except the date line, return address, closing, and writer's identification. These begin at the center of the line. (The center of the line is not a fixed point, it is calculated for the current line length.)

- The semiblocked format is the same as the blocked format; except, the first line of each paragraph is indented five spaces.
- In full-blocked format all lines begin at the left margin. There are no exceptions.
- The simplified format is the same as the full-blocked format; except, the salutation is replaced by an all-capital subject line and is followed by an additional blank line, the closing is omitted, and the writer's identification is in all-capital letters on one line.

Table 1 presents a synopsis of the placement of business letter components for the four **.LT** letter formats and lists the macros (which are explained in detail below) that you use to format those components.

Macro & Function	Placement on Page By .LT Argument BL	SB	FB	SP
.WA/.WE Writer's Address	Center	Center	Left	Left
.LO CN [*arg*] Confidential Notation	Left	Left	Left	Left
.LO RN[*arg*] Reference Notation	Center	Center	Left	Left
.IA/.IE Inside Address	Left	Left	Left	Left
.LO AT [*arg*] Attention	Left	Left	Left	Left
.LO SA [*arg*] Salutation	Left	Left	Left	None
.LO SJ [*arg*] Subject Line	Left	Indented	Left	Left
.P Paragraphs	Left	Indented	Left	Left
.FC Closing	Center	Center	Left	Left
.SG Signature	Center	Center	Left	Left
.NS/.NE Copy Notation	Left	Left	Left	Left

Figure 10: Arguments to the **.LT** Macro and their Functions

There are two possible error conditions for the **.LT** macro:

- If you omit the **.LT** macro, file processing aborts and an appropriate error message prints.

- If **mm** does not recognize an argument to **.LT**, the file processing aborts and an appropriate error message prints.

4.5.2. Writer's Address Macros (.WA, .WE)

Use this macro pair to specify the writer (author) of the letter and the writer's return address.

```
.WA writer-name [title]
Return address
.WE
```

For example,

```
.WA "James Lorrin, Ph.D." Director
Summit Research Company
38 River Road
Summit, New Jersey 07901
.WE
```

If a complete return address is not necessary for the letter (for example, if you use printed letterhead stationery) you can specify the writer information alone:

```
.WA "James Lorrin, Ph.D." Director
.WE
```

The return address cannot exceed 14 lines. Lines in the return address that follow line 14 do not appear on the letter.

The two arguments specified for the **.WA** and **.WE** macro pair, the *writer-name* and the *title,* provide information used by the **.SG** macro {5.1}. If you do not specify the **.SG** macro, the writer's name does not appear on the letter.

For the case of multiple writers on a single letter, you may specify only one writer return address. The specified writer return address must appear with the first writer-name as the first invocation of the **.WA/.WE** macro pair. Later return address specifications do not appear on the letter, although any number of additional writer names may be specified. For example,

```
.WA "James Lorrin, Ph.D." Director
Summit Research Company
38 River Road
Summit, New Jersey 07901
.WE
.WA "John Smith" Supervisor
.WE
.WA "Diane Kane" "Technical Support"
.WE
```

For blocked and semiblocked letter styles the writer return address begins on line 12 from the top of the first page and each line begins at the center of the line. For the full-blocked and simplified letter styles the writer return address begins on line 12 from the top of the page and each line begins at the left margin.

Top-of-page processing can be controlled directly through **nroff** or **troff**. The beginning of the printed page is user-defined. See the requests **.wh** and **.ch** in the "**nroff/troff** Technical Discussion."

If you omit either or both of the **.WA** and **.WE** macros, the file processing aborts and an appropriate error message prints.

4.5.3. Inside Address Macros (.IA, .IE)

.IA and **.IE** are a macro pair you use to specify the addressee and the addressee's address. There are two different ways that you can use this macro pair:

```
.IA
Text
.IE
```

or

```
.IA [addressee-name [title] ]
Text
.IE
```

For example,

```
.IA
Fred Smith, Ph.D.
Columbia University
116th Street
New York, New York 10019
.IE
```

or

```
.IA "Fred Smith, Ph.D."
.IE
```

For all four letter styles of **.LT**, the inside address prints on the fifth line below the date (if a reference notation or confidential notation appear after the date, the inside address prints three lines below the notation) and each line begins at the left margin.

If you omit either or both of the **.IA** and **.IE** macros, the file processing aborts and an appropriate error message prints.

4.5.4. Letter-Options Macro (.LO)

The letter-options macro provides the capability for specifying five common business letter components:

.LO *type* [*arg*]

The **.LO** macro takes care of placement and spacing of these letter components for each **.LT** letter format. **.LO** requires one argument to specify a letter component type, and accepts one optional string argument to refine its action.

Type	Corresponding Component
CN	confidential notation
RN	reference notation
AT	attention
SA	salutation
SJ	subject line

Figure 11: Types Given to **.LO** and their Functions

4.5.4.1. Confidential Notation (CN)

The confidential notation shows that a business letter should be read only by the person to whom it is addressed. The confidential notation appears on the second line below the date line of the letter and begins at the left margin for all letter formats.

If the optional string argument is present the specified string replaces the default. For example,

```
.LO CN "RESTRICTED"
```

The default of **CN** prints CONFIDENTIAL in upper-case.

4.5.4.2. Reference Notation (RN)

The reference notation supplies specific information to be used by the addressee. For example,

```
.LO RN "Meeting of 1/25"
```

The reference note appears two lines below the date line of the letter (or on the second line below any notation that follows the date) left aligned with the date line for all four letter formats.

RN provides a common format for including a reference note by printing the string **In reference to:** preceding the optional string argument to **.LO**. The format string **In reference to:** cannot be redefined. There is not a default value for the optional argument.

4.5.4.3. Attention (AT)

The attention line directs the letter to the attention of a specific person or department. For example,

```
.LO AT "Dr. Smith"
```

The attention information appears on the second line below the inside address of the letter and begins at the left margin.

AT provides a common format for directing a letter to the attention of a specific person by printing the string "ATTENTION:" preceding the optional string argument to **.LO**. The format string "ATTENTION:" cannot be redefined. There is not a default value for the optional argument.

4.5.4.4. Salutation (SA)

The salutation specifies the letter's opening greeting. For the blocked, semiblocked, and full-blocked formats the salutation appears on the second line below the inside address (or on the second line below the attention line, if used). In the simplified letter format, the salutation is ignored.

The default of **SA** prints "To Whom It May Concern:" for the salutation. If the optional string argument is present the specified string will replace the default. For example,

```
.LO SA "Dear Dr. Smith"
```

4.5.4.5. Subject Line (SJ)

The subject line shows what the letter is about. In the blocked and full-blocked letter formats the subject line information appears on the second line below the salutation and begin at the left margin. For the semiblocked format the subject line appears on the second line below the salutation and is indented five spaces. In the simplified letter format the subject line information appears in place of the salutation three lines below the inside address of the attention line; the salutation, if you use it, is ignored.

For the blocked, semiblocked, and full-blocked formats, **SJ** provides a common format for indicating what the letter is about by printing the string "SUBJECT: preceding the optional string argument to **.LO**.

```
.LO SJ "Staff Meeting:"
```

The format string "SUBJECT: cannot be redefined. There is not a default value for the optional argument.

For the simplified letter, the subject line string argument prints on the third line below the inside address or the attention line (a salutation is ignored if used).

If you specify the **.LO** macro without an argument or the argument you specify is unrecognized, the file processing aborts and an appropriate error message prints.

4.5.5. Multi-page Letters

The **.LT** macro controls the format for the first page of the letter. The letter macros will not alter the default **nroff/troff** page processing following the first page of the letter.

4.5.6. Sequence of Beginning Letter Macros

Macros **.WA, .WE, .IA, .IE,** and **.LT** must be given in the order listed in the following table. **.LO** can be specified multiple times with different argument *types*. The **.LO** argument *types* do not have to be in any specific order. All **.LO** requests must be specified before **.LT**.

```
.ND new date
.WA writer's name [title]
Return address
Street
City, State  Zip Code
Text
.WE
.IA
Addressee name
Title
Company
Street
City, State  Zip Code
Text
.IE
.LO type [arg]
.LT [arg]
.P
Text
.FC
.SG [arg [1] ]
.NS [arg [1] ]
Text
.NE
```

If you put **nroff/troff** requests and lines of text before **.LT**, you change how **.LT** works. For example, if the first line of a file is a line of text, **mm** processes the file as if you had not specified **.LT**.

5. Formatting the End of a Document

5.1. Formal Closing and Signature Line (.FC, .SG)

If you like, you may make your formal memorandum more formal with the **.FC** macro, which prints "Yours very truly," as a formal closing. You may specify an argument to **.FC**. to present a different closing.

The **.SG** (for "signature line") macro prints the author's name(s) after the formal closing, otherwise the author's name appears after the last line of the body of the memo. Each printed name begins at the center of the page. Three blank lines are left above each name for an author's signature. Use **.FC** before **.SG** or the formal closing will appear after the signature line.

Here's how you use **.FC** and **.SG** together. You may use either macro by itself.

```
.FC ["closing argument"]
.SG [arg [1] ]
```

You append a line of reference data including location code (for example, XF), department number (5414) and author's initials (JJS) by typing **.SG ""** instead of **.SG**. These data appear after the author's name and before the "Copy to" list (which is described below) and look like this:

```
XF-5414-JJS
```

If you type any other argument after **.SG** (not ""), that argument is treated as the typist's initials and is appended to the reference data. For example, if you specify:

```
.SG abc
```

the following line appears after the author's signature and before the "Copy to" list:

```
XF-5414-JJS-abc
```

If there are several authors and the second argument is given, reference data is placed on the signature line of the first author. This reference data contains only the location and department number of the first author, even though the other authors' initials are printed. Therefore, if there are authors from different departments and/or from different locations, the reference data should be supplied manually after calling **.SG** without arguments, like this:

```
.SG
XF-5414-JJS
XF-5415-SJF
```

This manually supplied information appears on the signature line of the second author.

For business letter style documents, the **.SG** macro prints the writer's name(s) after the formal closing, if any. Placement of the writer's names(s) for the signature block is controlled by the **.LT** macro specification (for example at the left margin or at the center of the page). The optional arguments accepted by **.SG** will not alter the processing of the macro when used in conjunction with **.LT**.

5.2. Approval Line (.AV)

Use the **.AV** macro after the last notation block to generate automatically a line for the approver's signature and the date.

.AV *approver's-name* [1]

For example,

```
.AV "Todd Doe"
```

produces

```
APPROVED:


_____________________________    _____________
Todd Doe                         Date
```

Use the optional second argument to prevent the mark "APPROVED:" from appearing above the approval line.

The **.SG** and **.NS** macros format signatures of authors and a list of notations, respectively. These macros do not work with released-paper style (**.MT 4**) {4.1.1}.

5.3. "Copy to" and Other Notations (.NS, .NE)

Many types of notations (such as a list of attachments or "Copy to" lists) may follow signature and reference data. Various notations are obtained through the **.NS** macro, which provides for proper spacing and for breaking notations across pages if necessary. You use **.NS** like this:

```
.NS [argument [1] ]
Names or other notation
.NE
```

If you use the optional second argument, the first argument will be used as the entire notation string. Codes for argument and the corresponding notations are

Argument	Notations
none	**Copy to**
	Copy to
0	**Copy to**
1	**Copy (with att.) to**
2	**Copy (without att.) to**
3	**Att.**
4	**Atts.**
5	**Enc.**
6	**Encs.**
7	**Under Separate Cover**
8	**Letter to**
9	**Memorandum to**
10	**Copy (with atts.) to**
11	**Copy (without atts.) to**
12	**Abstract Only to**
13	**Complete Memorandum to**
"string"	**Copy (string) to**
"string", with 2nd arg	**string**

Figure 12: Arguments to the **.NS** Macro

The *string*, which you may or may not enclose in double quotation marks, is placed within parentheses between the words "Copy" and "to" if it consists of more than two characters; otherwise, it is ignored. For example,

```
.NS "with att. 1 only"
```

generates

```
Copy (with att. 1 only) to
```

as the notation.

You may specify more than one notation before the **.NE** macro because a **.NS** macro stops the preceding notation, if any. For example,

```
.NS 4
Attachment 1-List of branch offices and managers
Attachment 2-List of regional offices and managers
.NS 1
Bill Taylor
.NS 2
J. Craven
A. Greenland
.NE
```

produces

```
Atts.
Attachment 1-List of branch offices and managers
Attachment 2-List of regional offices and managers

Copy (with att.) to
Bill Taylor

Copy (without att.) to
J. Craven
A. Greenland
```

If you use a second argument, the first argument becomes the entire notation. For example,

```
.NS "Table of Contents to" 1
```

produces

```
Table of Contents to
```

as the notation.

You may also use **.NS** and **.NE** at the beginning of a document following **.AS 2** and **.AE** to place the notation list on the Memorandum for File cover sheet {5.6}. If you give notations at the beginning without **.AS 2**, they will be saved and printed at the end of the document.

5.4. Table of Contents (.TC)

For most documents, the table of contents appears at the beginning, but because you have to call **.TC** at the end of your unformatted file, this Technical Discussion treats it as a macro for the end of the document. **mm** produces the table of contents at the end because the entire document must be processed before the table of contents can be generated from gathered section headings.

This macro normally appears once at the end of the document, after the Signature Block {5.1} and Notations {5.3} macros.

The **.TC** macro generates a table of contents containing heading levels that were saved for the table of contents as determined by the value of the **CI** register {3.5.3}.

.TC [*slevel*] [*spacing*] [*tlevel*] [*tab*] [*h1*] [*h2*] [*h3*] [*h4*] [*h5*]

Arguments to **.TC** control spacing before each entry, placement of associated page number, and additional text on the first page of the table of contents before the word "CONTENTS."

Spacing before each entry is controlled by the first and second arguments (*slevel* and *spacing*). Headings whose levels are less than or equal to *slevel* will have *spacing* blank lines (halves of a vertical space) before them. Both *slevel* and *spacing* default to 1. This means that first-level headings are preceded by one blank line (one-half a vertical space). The *slevel* argument does not control what levels of heading have been saved; saving of headings is the function of the **CI** register.

The third and fourth arguments (*tlevel* and *tab*) control placement of the associated page number for each heading. Page numbers can be justified at the right margin with either blanks or dots (called leaders) separating the heading text from the page number, or the page numbers can follow the heading text.

- For headings whose level is less than or equal to *tlevel* (default 2), page numbers are justified at the right margin. Here, the value of *tab* determines the character used to separate heading text from page number. If *tab* is 0 (default value), dots (that is, leaders) are used. If *tab* is greater than 0, spaces are used.

- For headings whose level is greater than *tlevel*, page numbers are separated from heading text by two spaces (that is, page numbers are "ragged right," not right justified).

Additional arguments (*h1 . . . h5*) are horizontally centered on the page and precede the table of contents.

5.5. Changing the Table of Contents (.TX, .TY)

If you call the **.TC** macro with at most four arguments, **mm** calls the user-exit macro **.TX** (without arguments) before the word "CONTENTS" is printed, or calls the user-exit macro **.TY**, and does not print the word "CONTENTS." By defining **.TX** or **.TY** and calling **.TC** with at most four arguments, you can specify what needs to be done at the top of the first page of the table of contents. For example,

```
.de TX
.ce 2
Special Application
Message Transmission
.sp 2
.in +5n
Approved: \l'2.5i'
.in
.sp
..
.TC
```

yields the following output when you format the file that contains them:

```
                        Special Application
                        Message Transmission


Approved: _____________________________________
```

If you define the **.TX** macro as **.TY**, the word "CONTENTS" would be suppressed. Defining **.TY** as an empty macro suppresses "CONTENTS" with no replacement:

```
.de TY
..
```

By default, the first level headings appear in the table of contents left justified. Later levels align with the text of headings at the preceding level. You may change these indentations by defining the **Ci** string that takes a maximum of seven arguments corresponding to the heading levels. You must give at least as many arguments as are set by the **Cl** register. With these registers, arguments must be scaled. For example, with "**Cl** = 5":

```
.ds Ci .25i .5i .75i 1i 1i  \" troff
```

or

```
.ds Ci 0 2n 4n 6n 8n  \" nroff
```

Two other registers are available to change the format of the table of contents:**Oc** and **Cp**.

- By default, table of contents pages will have lower-case roman numeral page numbering. If the **Oc** register is set to 1, the **.TC** macro will not print any page number but will instead reset the **P** register to 1. It is your responsibility to give an appropriate page footer to specify the placement of the page number. Ordinarily, the same **.PF** macro (page footer) used in the body of the document is adequate.
- Front matter pages, such as those listing figures and tables, will be produced separately unless **Cp** is set to 1, which causes these lists to appear on the same page as the table of contents.

5.6. Cover Sheet (.CS)

Like the table of contents macro, you call the cover sheet macro (**.CS**) at the end of the document, even though you usually put a cover sheet at the beginning. **.CS** generates a cover sheet in either the released paper or memorandum type of the formal memorandum style.

.CS [*pages* [*other* [*total* [*figs* [*tbls* [*refs*]]]]]]

All other information for the cover sheet is obtained from data given before the **.MT** macro call {4.1.1}. If you use the memorandum type, the **.CS** macro generates the "Cover Sheet for Technical Memorandum". The data that appear in the lower left corner of the memorandum cover sheet (counts of: pages of text, other pages, total pages, figures, tables, and references) are generated automatically (0 is used for "other pages"). You may change these values by supplying the corresponding arguments to the **.CS** macro. If you use the released-paper style, all arguments to **.CS** are

ignored.

5.7. References (.RS, .RF, .RP)

Obtain automatically numbered references by typing ***(Rf** (invoking the string **Rf**) immediately after the text you want to reference. This places the next sequential reference number (in a smaller point size) enclosed in brackets one-half line above the text you reference. **mm** keeps reference count in the **Rf** number register. **mm** uses the number register **:R** to print the reference number for each reference call in the text (***(Rf**). You may change the format or value of the **:R** register to effect the reference marks, without affecting the total count of references.

Use the **.RS** and **.RF** macros to delimit text of each reference.

```
Text to be referenced.\*(Rf
.RS
Reference
.RF
```

The **.RS** macro takes an optional argument, a *string-name*. For example,

```
.RS aA
Reference
.RF
```

The string **aA** is assigned the current reference number. You may use this string later in the document as the string call, ***(aA**, to reference text that must be labeled with a prior reference number. The reference is enclosed in brackets one-half line above the text to be referenced. You do not need a **.RS/.RF** pair for subsequent references.

You may use the **.RP** (reference page) macro to produce reference pages anywhere else within a document (that is, after each major section). It is not needed to produce a separate reference page with default spacings at the end of the document.

The **.RP** macro produces the reference page.

```
.RP [arg1 [arg2] ]
```

Two arguments may be used with **.RP**. Possibilities for the first argument are as follows:

Argument 1	Function
0	Reset reference counter (default).
1	Do not set reference counter.

Possibilities for the second argument are as follows:

Argument 2	Function
0	Put on separate page (default).
1	Do not cause a following **.SK**.
2	Do not cause a preceding **.SK**.
3	Do not cause a preceding or following **.SK**.

This page contains the reference items (that is, reference text enclosed within **.RS/.RF** pairs). Reference items are separated by a space (one-half a vertical space) unless you set the **Ls** register to 0 to suppress this spacing. You may change the reference page title by defining the **Rp** string:

```
.ds Rp "New Title"
```

If no **.SK** macro is issued by the **.RP** macro, a single blank line separates the references from the following/preceding text. You may wish to adjust spacing. For example, to produce references at the end of each major section:

```
.sp 3
.RP 1 2
.H 1 "Next Section"
```

6. Miscellaneous Macros

6.1. Bold, Italic, and Roman Fonts (.B, .I, .R)

mm provides three macros for changing the prevailing font of a document.

```
.B [bold-arg [previous-font-arg] ] ...
.I [italic-arg [previous-font-arg] ] ...
.R
```

When you invoke it without arguments, **.B** changes the font to bold, and **.I** changes to underlining (italic). **mm** uses these fonts until you invoke another font macro, such as the **.R** macro, which restores roman font. Thus,

```
.I
here is some text.
.R
```

yields underlined text via the **nroff** and italic text via the **troff** formatter.

If you invoke the **.B** or **.I** macro with one argument, that argument appears in the appropriate font (underlined in the **nroff** formatter for **.I**). Then **mm** restores the previous font (underlining is turned off in the **nroff** formatter). If you give two or more arguments (maximum six) with a **.B** or **.I** macro call, the second argument is concatenated to the first with no intervening space (1/12 space if the first font is italic), but it is printed in the previous font. Remaining pairs of arguments are similarly alternated. For example,

```
.I italic " words of text " right-justified
```

produces

italic words of text *right-justified*

The **.B** and **.I** macros alternate with the prevailing font at the time the macros are invoked. To alternate specific pairs of fonts, the following macros are available:

```
.IB   .BI   .IR   .RI   .RB   .BR
```

Each macro takes a maximum of six arguments and alternates arguments between specified fonts.

When you use a terminal that cannot underline, you can insert the following line in the parameter setting segment to eliminate all underlining:

```
.rm ul
.rm cu
```

mm handles font changes in headings separately {3.5.2.4}.

6.2. Justification of Right Margin (.SA)

Use the **.SA** macro to set right-margin justification for the main body of text.

.SA [*arg*]

Choose from two justification flags: *current* and *default*. Initially, **mm** sets both flags for no justification in the **nroff** formatter and for justification in the **troff** formatter. An argument to **.SA** causes the following action:

Argument	Meaning
0	Sets both flags to no justification. It acts like the **.na** request.
1	Sets both flags to cause both right and left justification, the same as the **.ad** request.
Omitted	Causes the current flag to be copied from the default flag, thus performing either a **.na** or **.ad** depending on the default condition.

Figure 13: Arguments to the **.SA** Macro

In general, you can use the no adjust request (**.na**) to ensure that justification is turned off. Use **.SA** to restore justification, rather than the **.ad** request. Specify justification or no justification for the remainder of the text by inserting **.SA 1** or **.SA 0** one time at the parameter setting segment.

6.3. Skipping Pages (.SK)

The **.SK** macro skips pages but retains the usual header and footer processing.

```
.SK [pages]
```

If you omit the *pages* argument, or make it null or 0, **.SK** skips to the top of the next page unless it is currently at the top of a page (then it does nothing). **.SK** *n* skips *n* pages. **.SK** positions text that follows it at the top of a page, while **.SK 1** leaves one page blank except for the header and footer.

6.4. Forcing an Odd Page (.OP)

Use the **.OP** macro to ensure that formatted output text following the macro begins at the top of an odd-numbered page.

```
.OP
```

- If currently at the top of an odd-numbered page, text output begins on that page (no motion takes place).
- If currently on an even page, text resumes printing at the top of the next page.
- If currently on an odd page (but not at the top of the page), one blank page is produced, and printing resumes on the next odd-numbered page after that.

6.5. Setting Point Size and Vertical Spacing (.S)

Change the prevailing point size and vertical spacing by invoking the **.S** macro.

.S [*point size* [*vertical_spacing*]]

In the **troff** formatter, the default point size (obtained from the **mm** register *S* {8.4}) is 10 points, and the vertical spacing is 12 points (six lines per inch). You may use the mnemonics D (default value), C (current value), and P (previous value) for both arguments.

- If an argument is *negative*, current value is decremented by the specified amount.
- If an argument is *positive*, current value is incremented by the specified amount.
- If an argument is *unsigned*, it becomes the new value.
- If there are no arguments, the **.S** macro defaults to P.
- If you specify the first argument but not the second, then (default) D is used for the vertical spacing.

Default value for vertical spacing is always two points greater than the current point size. Footnotes {3.3} are two points smaller than the body with an additional 3-point space between footnotes. A null ("") value for either argument defaults to C (current value). Thus, if *n* is a numeric value:

Macro and Argument	Result
.S	.S P P
.S *n*	.S C n
.S *n*	.S n C
.S *n*	.S n D
.S	.S C D
.S	.S C C
.S *n n*	.S n n

Figure 14: Arguments to the **.S** Macro

P means previous value, **D** means default value (usually 10 point), and **C** means

current value

If you make the first argument greater than 99, the default point size (10 points) is restored. If the second argument is greater than 99, **mm** uses the default vertical spacing (current point size plus two points). For example,

.S 100 = .S 10 12
.S 14 111 = .S 14 16

The **.SM** macro allows you to reduce by one point the size of a string.

```
.SM string1 [string2] [string3]
```

If you omit the third argument (*string3*), the first argument (*string1*) is made smaller and is concatenated with the second argument (*string2*), if specified. If all three arguments are present (even if any are null), the second argument is made smaller and all three arguments are concatenated. For example,

Input	Output
.SM X	**X**
.SM X Y	**XY**
.SM Y X Y	**YXY**
.SM YXYX	**YXYX**
.SM YXYX)	**YXYX)**
.SM (YXYX)	**(YXYX)**
.SM Y XYX ""	**YXYX**

6.6. Producing Accents

The following example shows that you may use strings to produce accents for letters:

Name	Input	Output
Grave accent	**c***	**c**
Acute accent	**e*'**	**é**
Circumflex	**o*^**	**ô**
Tilde	**n*~**	**ñ**
Cedilla	**c*,**	**ç**
Lower-case umlaut	**u*:**	**ü**
Upper-case umlaut	**U*;**	**Ü**

6.7. Two-Column Output (.2C, .1C)

The **mm** text formatting macro package can format two-columns on a page. The **.2C** macro begins 2-column processing that continues until a **.1C** macro (1-column processing) is encountered.

```
.2C
text and formatting requests (except another .2C)
.1C
```

In 2-column processing, each physical page is thought of as containing 2-columnar "pages" of equal (but smaller) "page" width. Page headers and footers are not affected by 2-column processing. The **.2C** macro does not balance 2-column output.

It is possible to have full-page width footnotes and displays when in 2-column mode though footnotes and displays are by default narrow in 2-column mode and wide in 1-column mode. **.WC** controls footnote and display width, which is specified in arguments. **.WC WD FF**, for instance, causes all displays to be wide and all footnotes to be the same width. **.WC N** reinstates the defaults. If you give conflicting settings to **.WC**, **mm** selects the last one: a **.WC WF −WF** command has the effect of a **.WC −WF**. Arguments to **.WC** are as follows:

Argument:	Meaning:
N	Default mode (**−WF**, **−FF**, **−WD**, **FB**).

WF	Wide footnotes (even in 2-column mode).
–WF	DEFAULT: Turn off **WF**. Footnotes follow column mode; wide in 1-column mode (**.1C**), narrow in 2-column mode (**.2C**), unless **FF** is set.
FF	First footnote. All footnotes have same width as first footnote encountered for that page.
–FF	DEFAULT: Turn off **FF**. Footnote style follows settings of **WF** or **–WF**.
WD	Wide displays (even in 2-column mode).
–WD	DEFAULT: Displays follow the column mode in effect when display is encountered.
FB	DEFAULT: Floating displays cause a break when output on the current page.
–FB	Floating displays on current page do not cause a break.

6.8. Inserting Text Interactively (.RD)

The **.RD** (read insertion) macro allows you to stop the standard output of a document and to read text from the standard input until two consecutive newline characters are found.

.RD [*prompt* [*diversion* [*string*]]]

When newline characters are encountered, normal output is resumed.

- The *prompt* argument prints at the terminal. If not given, **.RD** signals you with a BEL on terminal output.
- The *diversion* argument allows you to save all text typed in after the prompt in a macro whose name is that of the diversion.
- The *string* argument allows you to save for later reference the first line following the prompt in the named string.

The **.RD** macro follows the formatting conventions in effect. Thus, the following examples assume that the **.RD** is invoked with line-filling off (**.nf**):

```
.RD Name aA bB
```

produces

```
Name: J. Jones    (you type name)
16 Elm Rd.,
Piscataway
```

The diverted macro **.aA** will contain

```
J. Jones
16 Elm Rd.,
Piscataway
```

The string **bB** (***(bB**) contains "J. Jones."

A newline character followed by an EOF (user specifiable CONTROL-d) also allows you to resume normal output.

6.9. SCCS Release Identification (RE)

The **RE** string contains the SCCS release and the **mm** text formatting macro package current version level. For example,

```
This is version \*(RE of the macros.
```

produces

```
                        - 1 -


This is version 10.129 of the macros.
```

This information is useful in analyzing suspected bugs in **mm**. The easiest way to have the release identification number appear in the output is to specify **−rD1** {8.4} on the command line. This causes the **RE** string to be output as part of the page header {3.4.1}.

6.10. Extending and Changing the Macros

6.10.1. Naming Conventions

In this part, the following conventions are used to describe names:

n Digit

a Lower-case letter

A Upper-case letter

x Any alphanumeric character (n, a, or A, that is, letter or digit)

s Any nonalphanumeric character (special character)

Request, macro, and string names are kept by the formatters in a single internal table; therefore, there must be no duplication among such names. Number register names are kept in a separate table.

6.10.1.1. Components Used by Formatters

Requests: *aa* (most common)
an (only one, currently: c2)

Registers: *aa* (normal)
.x (normal)
.s (only one, currently: .$)
a. (only one, currently: c.)

6.10.1.2. Components Used by mm

Macros and Strings: *A*, *AA*, *Aa* (accessible to you; for example, macros **.P** and **.HU**, strings **F**, **BU**, and **Lt**)

nA (accessible to you; only two, currently: **1C** and **2C**)

aA (accessible to you; only one, currently: **nP**)

s (accessible to you; only the seven accents, currently {6.6})

registers: *An*, *Aa* (accessible to you; for example, **H1**, **Fg**)

A (accessible to you; meant to be set on the command line; for example, **C**)

6.10.2. Sample Extensions

6.10.2.1. Appendix Headings

The following is a way of generating and numbering appendix headings:

```
.nr Hu 1
.nr a 0
.de aH
.nr a +1
.nr P 0
.PH "''''Appendix \\na-\\\\nP''"
.SK
.HU "\\$1"
..
```

After the above initialization and definition, each call of the form

.aH "*Title*"

begins a new page (with the page header changed to "Appendix *a–n*") and generates an unnumbered heading of *title*, which, if desired, can be saved for the table of contents. To center appendix titles the **Hc** register must be set to 1 {3.5.2.3}.

6.10.2.2. Hanging Indent with Tabs.

The following example uses the hanging indent feature of variable-item lists {3.2.4}. A user-defined macro is defined to accept four arguments that make up the *mark*. In the output, each argument is to be separated from the previous one by a tab; tab settings are defined later. Since the first argument may begin with a period or apostrophe, the **\&** is used so that the formatter does not interpret such a line as a formatter request or macro call. The 2-character sequence **\&** is understood by formatters to be a zero-width space. It causes no output characters to appear, but it removes the special meaning of a leading period or apostrophe. The **\t** is translated by the formatter into a tab. The **\c** is used to concatenate the input text that follows the macro call to the line built by the macro.

```
.de aX
.LI
\&\\$1\t\\$2\t\\$3\t\\$4\t\c
..
.ta 8 14 20 24
.VL 1.0i
.aX .nh off \- no
No hyphenation.
Automatic hyphenation is turned off.
Words containing hyphens
(for example, mother-in-law) may still be split across lines.
.aX .hy on \- no
Hyphenate.
Automatic hyphenation is turned on.
.aX .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.
.LE
```

The resulting output is:

```
                              - 1 -


No hyphenation. Automatic hyphenation is turned off.  Words
 containing hyphens (for example, mother-in-law) may still
 be split across lines.

Hyphenate.  Automatic hyphenation is turned on.

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.
```

6.11. Vertical Spacing (.SP)

There exists several ways of obtaining vertical spacing, all with different effects. The **.sp** request spaces the number of lines you specify unless you turn off spacing with **.ns**, then **mm** ignores the **.sp** request. You can restore spacing with the **.rs** request.

Spacing is turned off automatically at the end of a page header to eliminate spacing by a **.sp** or **.bp** request that happens to occur at the top of a page.

Use the **.SP** macro to avoid the accumulation of vertical space by successive macro calls.

```
.SP [lines]
```

Several **.SP** calls in a row will not produce the sum of the arguments but only the maximum argument. For example, the following produces only three blank lines:

```
.SP 2
.SP 3
.SP
```

Many **mm** macros use **.SP** for spacing. For example, if you immediately follow **.LE 1** {3.2.1} with **.P** {3.1}, **mm** produces only a single blank line (one-half vertical space) between the end of the list and the following paragraph. An omitted argument to **.SP** defaults to one blank line (one vertical space). Negative arguments are not permitted. The argument must be unscaled, but fractional amounts are permitted.

The **.ns** request also inhibits the **.SP** macro (as well as the **.sp** request).

7. Troubleshooting and Error Messages

7.1. Error Terminations

When an **mm** macro detects an error, it performs the following actions:

- It performs a line break.
- It prints the formatter output buffer (which may contain some text) to avoid confusion regarding location of the error.
- It prints a short message giving the name of the macro that detected the error, type of error, and approximate line number in the current input file of the last processed input line. The last section of this chapter explains error messages.
- It stops processing unless register **D** {8.4} has a positive value. In the latter case, it continues processing even though the output is guaranteed to be garbled from that point on.

The error message outputs directly to your terminal. If you are using an output filter, such as **300**, **450**, or **hp** to postprocess the **nroff** formatter output, this message may be garbled by being intermixed with text that is held in that filter's output buffer. If you are using any **eqn/neqn** or **tbl** material, and if the **−o***list* option of the formatter causes the last page of the document not to be printed, a harmless "broken pipe" message may result.

7.2. Disappearance of Output

Disappearance of output usually occurs because of an unclosed diversion (for example, a missing **.DE** or **.FE** macro). Fortunately, macros that use diversions are careful about it, and these macros check to make sure that illegal nestings do not occur. If any error message is issued about a missing **.DE** or **.FE**, search backwards from the termination point and look for the corresponding **.DF**, **.DS**, or **.FS** (since you must use these macros in pairs).

The command **grep −n '^\.[EDFRT] [EFNQS]'** *files* ... prints all the **.DF**, **.DS**, **.DE**, **.EQ**, **.EN**, **.FS**, **.FE**, **.RS**, **.RF**, **.TS**, and **.TE** macros found in *files* ..., each preceded by its file name and the line number in that file. Use this listing to check for illegal nesting and/or omission of these macros. Also, the **checkmm** programs finds mismatched macro pairs.

7.3. Error Messages

An **mm** error message has a standard part followed by a variable part. The standard part has the form:

ERROR: (*file*)input line *n*:

Variable parts consist of a descriptive message usually beginning with a macro name. Appendix D lists them in alphabetical order by macro name, each with a more complete explanation.

8. Using the mm Command Line

Use the **mm** command line to print documents using **nroff** and the **mm** macros. Use **mmt** to format documents with **troff** and the macros.

8.1. Typical mm Command Lines and Options

These are the typical command lines for **mm** (*options* are described below). Pipe results to a printer or phototypesetter as appropriate, or redirect output to a file.

- Text without tables or equations:

  mm [*options*] *file* ...

  mmt *options file* ...

- Text with tables:

  mm −t *options file* ...

  mmt −t *options file* ...

- Text with equations:

  mm −e *options file* ...

  mmt −e *options file* ...

- Text with line pictures:

  mmt −p *options file* ...

- Text with graphs:

  mmt −g *options file* ...

- Text with both tables and equations:

  mm −t −e *options file ...*

  mmt −t −e *options file ...*

- Text with tables, equations, pictures and graphs:

  mmt −t −e −p −g *options file ...*

You can put *options* in any order you wish, but you must put them before the *file* name(s).

−e calls **neqn (mm)** or **eqn (mmt)** and causes it to read **/usr/pub/eqnchar**

−c calls **col**(1)

−t calls **tbl**

−p calls **pic** (use this with **mmt** only)

−g calls **grap** (use this with **mmt** only)

−E calls the **−e** option of **nroff**

−12 specifies that 12 characters per inch, or 78 characters per normal output line (6 1/2 inches) are output. This specification is called "12-pitch mode." The pitch switch on your printer should be set to 12. By default, 10 characters per inch are output.

−T*tty_type*
specifies to which output device you are sending your output.

When you format a document, prepare the output for a particular type of terminal, because the output may require features that are specific to that terminal. For example, some terminals produce reverse paper motion or half-line paper motion in both directions. A list of the supported terminals you may use with **mm** appears in the following section. (Check with your system administrator for a list of locally supported devices.)

If you use two-column processing {6.7}, you must specify either the **−c** option to **mm** [**mm** uses **col**(1) automatically for many terminal types] or you must postprocess **nroff** formatter output with **col**(1). In the latter case, you must specify the **−T37** terminal type to the **nroff** formatter, take care not to specify the **−h** option, and process the output of **col**(1) by the appropriate terminal filter (for example, **450**); **mm** with the

-c option handles all this automatically.

8.2. Command Line Options for Specifying a Printer

Use one of the following arguments to the **mm** command line option **-T** to specify an output device to the formatter. To specify a program that drives the printer, use the **-T** option. It is a way of telling the formatter to carry out certain operations that your particular printer program can execute. Ask your system administrator what argument to **-T** is appropriate for your text formatting set-up. A list of all supported arguments (values for *tty_type*) and the devices they signify are as follows:

-T*tty_type*

2631	Hewlett-Packard 2631 printer in regular mode
2631-c	Hewlett-Packard 2631 printer in compressed mode
2631-e	Hewlett-Packard 2631 printer in expanded mode
300	DASI-300 printer
300-12	DASI-300 terminal set to 12-pitch (12 characters per inch)
300s	DASI-300s printer (**300S** is a synonym)
300s-12	DASI-300s printer set to 12-pitch (12 characters per inch) (**300S-12** is a synonym)
37	Teletype Model 37 terminal
382	DTC-382
4000a	Trendata 4000a terminal (**4000A** is a synonym)
450	DASI-450 (Diablo Hyterm) printer
450-12	DASI-450 terminal set to 12-pitch (12 characters per inch)
832	Anderson Jacobson 832 terminal
8510	C.ITOH printer
lp	generic name for printers that can underline and tab. (All text using reverse linefeeds, such as those having tables, that is sent to **lp** must be processed with **col.**) **lp** is the default device for **mm.**
tn300	GE Terminet 300 terminal
X	printers equipped with TX print train

8.3. Giving nroff or troff the −mm Flag

You may also use the **mm** package by including the **−mm** flag as an argument to **nroff** or **troff**. The **−mm** flag causes the file **/usr/lib/tmac/tmac.m** to be read and processed before any other files. This action:

- defines the Memorandum Macros
- sets default values for various parameters
- initializes the formatter to be ready to process input text files

8.4. Setting Number Registers from the Command Line

With **mm**, you commonly use number registers to hold parameter values that control various aspects of output style. You can change many of these values within the text files with **.nr** requests. In addition, you can set some of these registers from the command line. This is useful when parameters should not be embedded within the input text. The number register meanings are

−rA*n* *n* = 1 has effect of invoking the **.AF** macro without an argument {4.1.7}.

−rC*n* Sets type of copy (for example, DRAFT) to be printed at bottom of each page {3.4.3}.
n = 1 for OFFICIAL FILE COPY.
n = 2 for DATE FILE COPY.
n = 3 for DRAFT with single spacing and default paragraph style.
n = 4 for DRAFT with double spacing and 10-space paragraph indent.

−rD1 Sets *debug* mode.
This flag requests formatter to continue processing even if **mm** detects errors that would otherwise cause termination. It also includes some debugging information in the default page header {3.4.1}.

−rE*n* Controls font of Subject/Date/From fields.
n = 0, fields are bold (default for the **troff** formatter).
n = 1, fields are roman font (regular text-default for the **nroff** formatter).

−rL*k* Sets length of physical page to *k* lines.
For the **nroff** formatter, *k* is an unscaled number representing lines.
For the **troff** formatter, *k* must be scaled.
Default value is 66 lines per page.
This flag is used, for example, when directing output to a Versatec printer.

−rN*n* Specifies page numbering style (See Figure 8-1).
n = 0 (default), all pages get the prevailing header {3.4.1}.
n = 1, page header replaces footer on page 1 only.
n = 2, page header is omitted from page 1.
n = 3, "section-page" numbering {3.5.3} occurs (**.FD** {3.3.1} and **.RP** {5.7} defines footnote and reference numbering in sections).
n = 4, default page header is suppressed; however, a user-specified header is not affected.
n = 5, "section-page" and "section-figure" numbering occurs.

n	Page 1	Pages 2-*end*
0	header	header
1	header replaces footer	header
2	no header	header
3	"section-page" as footer	same as page 1
4	no header	no header unless .PH defined
5	"section-page" as footer and "section-figure"	same as page 1

Figure 15: Effects of the *n* Register on Page Numbering Style

Contents of the prevailing header and footer do not depend on number register *N* value; *N* controls whether the header (*N*=3) or the footer (*N*=5) is printed, as well as the page numbering style. If header and footer are null {3.4}, the value of *N* is irrelevant.

−rO*k* Offsets output *k* spaces to the right.
For the **nroff** formatter, *k* is an unscaled number representing lines or character positions.
For the **troff** formatter, *k* must be scaled.
This flag is helpful for adjusting output positioning on some terminals.
The default offset if this register is not set on the command line is

0.75 inch (**nroff**) and 0.5 inch (**troff**).

−rP*n* Specifies that pages of the document are to be numbered starting with *n*.
This register may also be set via a **.nr** request in the input text.

−rS*n* Sets point size and vertical spacing for the document.
The default *n* is 10, that is, 10-point type on 12-point vertical spacing, giving six lines per inch {6.5}.
This flag applies to the **troff** formatter only.

−rT*n* Provides register settings for certain devices.
If *n* is 1, line length and page offset are set to 80 and 3, respectively.
Setting *n* to 2 changes the page length to 84 lines per page and inhibits underlining; it is meant for output sent to the Versatec printer.
This flag applies to the **nroff** formatter only.

−rU1 Controls underlining of section headings. This flag causes only letters and digits to be underlined. Otherwise, all characters (including spaces) are underlined {3.5.2.4.2}. This flag applies to the **nroff** formatter only.

−rW*k* Sets page width (line length and title length) to *k*.
For the **nroff** formatter, *k* is an unscaled number representing character positions.
For the **troff** formatter, *k* must be scaled.
This flag can be used to change page width from the default value of 6 inches (60 characters in 10 pitch or 72 characters in 12 pitch).

9. Appendices

9.1. Appendix A: mm Macro Name Summary

The following listing shows all the **mm** macros and their usage. Each item in the list gives a definition of the macro followed by its normal format and arguments. The numbers enclosed in braces {} show the section in this technical discussion where a complete explanation of each macro may be found. You do not call macros marked with an asterisk directly. They are "user exits" that you define, and they are called by the **mm** macros from inside header, footer, or other macros.

.1C 1-column processing {6.7}
.1C

.2C 2-column processing {6.7}
.2C

.AE Abstract end {4.2.1}
.AE

.AF Alternate format of "subject:/date:/from:" block {4.1.7}
.AF ["*name of firm*"]

.AL Automatically incremented list start {3.2.2}
.AL [*type* [*text-indent* [1]]]

.AS Abstract start {4.2.1}
.AS [*arg* [*indent*]]

.AT Author's title {4.1.6}
.AT [*title*] . . .

.AU Author information {4.1.5}
.AU name [*initials* [*loc* [*dept* [*ext* [*room* [*arg* [*arg* [*arg*]]]]]]]]

.AV Approval signature {5.2}
.AV [*name* [1]]

.B bold {6.1}
.B [*bold-arg* [*previous-font-arg*]] . . .

.BE Bottom block end {4.2.3}
.BE

.BI bold/italic {6.1}
.BI [*bold-arg* [*italic-arg* [*bold* [*italic* [*bold* [*italic*]]]]]]

.BL Bullet list start {3.2.5}
.BL [*text-indent* [1]]

.BR bold/roman {6.1}
.BR [*bold-arg* [*roman-arg* [*bold* [*roman* [*bold* [*roman*]]]]]]

.BS Bottom block start {4.2.3}
.BS

.CS Cover sheet {5.6}
.CS [*pages* [*other* [*total* [*figs* [*tbls* [*refs*]]]]]]

.DE Display end {3.6.1}
.DE

.DF Display floating start {3.6.2}
.DF [*format* [*fill* [*rindent*]]]

.DL Dash list start {3.2.5}
.DL [*text-indent* [1]]

.DS Display static start {3.6.1}
.DS [*format* [*fill* [*rindent*]]]

.EC Equation caption {3.9}
.EC [*title* [*override* [*flag*]]]

.EF Even-page footer {3.4.4}
.EF [*arg*]

.EH Even-page header {3.4.2}
.EH [*arg*]

.EN End equation display {3.8}
.EN

.EQ Equation display start {3.8}
.EQ [*label*]

.EX Exhibit caption {3.9}
.EX [*title* [*override* [*flag*]]]

.FC Formal closing {5.1}
.FC [*closing*]

.FD Footnote default format {3.3.1}
.FD [*arg* [1]]

.FE Footnote end {3.3}
.FE

.FG Figure title {3.9}
.FG [*title* [*override* [*flag*]]]

.FS Footnote start {3.3}
.FS [*label*]

.H Heading—numbered {3.5}
.H level [*heading-text* [*heading-suffix*]]

.HC Hyphenation character {1.4.4}
.HC [*hyphenation-indicator*]

.HM Heading mark style (Arabic or roman numerals, or letters) {3.5.2.6}
.HM [*arg1*] . . . [*arg7*]

.HU Heading—unnumbered {3.5.1}
.HU *heading-text*

.HX* Heading user exit X (before printing heading) {3.5.4}
.HX *dlevel rlevel heading-text*

.HY* Heading user exit Y (before printing heading) {3.5.4}
.HY *dlevel rlevel heading-text*

.HZ* Heading user exit Z (after printing heading) {3.5.4}
.HZ *dlevel rlevel heading-text*

.I italic (underline in the **nroff** formatter) {6.1}
.I [*italic-arg* [*previous-font-arg*]] . . .

.IA Inside address start {4.5.3}
.IA [*addressee-name* [*title*]]

.IB italic/bold {6.1}
.IB [*italic-arg* [*bold-arg* [*italic* [*bold* [*italic* [*bold*]]]]]]

.IE Inside address end {4.5.3}
.IE

.IR italic/roman {6.1}
.IR [*italic-arg* [*roman-arg* [*italic* [*roman* [*italic* [*roman*]]]]]]

.LB List begin {3.2.7}
.LB *text-indent mark-indent pad type* [*mark* [*LI-space*]]
[LB-space]

.LC List-status clear {3.2.8}
.LC [*list-level*]

.LT Letter Type {4.5.1}
.LT [*arg*]

.LE List end {3.2.1}
.LE [1]

.LI List item {3.2.1}
.LI [*mark* [1]]

.LO Letter Options {4.5.4}
.LO *type* [*arg*]

.ML Marked list start {3.2.3}
.ML *mark* [*text-indent* [1]]

.MT Memorandum type {4.1.1}
.MT [*type* [*addressee*]] *or* .MT [4] [1]

.ND New date {4.1.3}
.ND *new-date*

.NE Notation end {5.3}
.NE

.NS Notation start {5.3}
.NS [*arg* [1]]

.nP Double-line indented paragraphs {3.1.2}
.nP

.OF Odd-page footer {3.4.4}
.OF [*arg*]

.OH Odd-page header {3.4.2}
.OH [*arg*]

.OK Other keywords for the Technical Memorandum cover sheet {4.2.2}
.OK [*keyword*] . . .

.OP Odd page {6.4}
.OP

.P Paragraph {3.1}
.P [*type*]

.PF Page footer {3.4.3}
.PF [*arg*]

.PH Page header {3.4.1}
.PH [*arg*]

.PM Proprietary marking {4.3}
.PM [*code*]

.PX* Page-header user exit {3.4.9}
.PX

.R Return to regular (roman) font {6.1}
.R

.RB roman/bold {6.1}
.RB [*roman-arg* [*bold-arg* [*roman* [*italic* [*roman* [*italic*]]]]]]

.RD Read insertion from terminal {6.8}
.RD [*prompt* [*diversion* [*string*]]]

.RF Reference end {5.7}
.RF

.RI roman/italic {6.1}
.RI [*roman-arg* [*italic-arg* [*roman* [*italic* [*roman* [*italic*]]]]]]

.RL Reference list start {3.2.5}
.RL [*text-indent* [1]]

.RP Produce reference page {5.7}
.RP [*arg* [*arg*]

.RS Reference start {5.7}
.RS [*string-name*]

.S Set **troff** formatter point size and vertical spacing {6.5}
.S [*size* [*spacing*]]

.SA Set adjustment (right-margin justification) default {6.2}
.SA [*arg*]

.SG Signature line {5.1}
.SG [*arg* [1]]

.SK Skip pages {6.3}
.SK [*pages*]

.SM Make a string smaller {6.5}
.SM *string1* [*string2* [*string3*]]

.SP Space vertically {6.11}
.SP [*lines*]

.TB Table title {3.9}
.TB [*title* [*override* [*flag*]]]

.TC Table of contents {5.4}
.TC [*slevel* [*spacing* [*tlevel* [*tab* [*h1* [*h2* [*h3* [*h4* [*h5*]]]]]]]]]

.TE Table end {3.7}
.TE

.TH Table header {3.7}
.TH [*N*]

.TL Title of memorandum {4.1.4}
.TL [*charging-case* [*filing-case*]]

.TM Technical Memorandum number(s) {4.1.2}
.TM [*number*] . . .

.TP* Top-of-page macro {3.4.9}
.TP

.TS Table start {3.7}
.TS [H]

.TX* Table of contents user exit {5.5}
.TX

.TY* Table of contents user exit (suppresses "CONTENTS") {5.4.1}
.TY

.VL Variable-item list start {3.2.4}
.VL *text-indent* [*mark-indent* [1]]

.VM Vertical margins {3.4.7}
.VM [*top* [*bottom*]]

.WA Writer's address start {4.5.2}
.WA *writer-name* [*title*]

.WC Footnote and display width control {6.7}
.WC [*format*]

.WE Writer's address end {4.5.2}
.WE

9.2. Appendix B: mm String Name Summary

The following list shows the predefined string names used by the **mm** macro package. The numbers in braces {} show the section in this technical discussion where more information about the string can be found.

BU Bullet {1.4.6}
nroff: ⊕
troff: •

Ci Table of contents indent list {5.4.1}
Up to seven scaled arguments for heading levels

DT Date {1.2}
Current date, unless overridden
Month, day, year (for example, May 31, 1979)

EM Em dash string {1.4.7}
Produces an em dash in the **troff** formatter and a double hyphen in **nroff**

F Footnote number generator {3.3}
nroff: \u\\n+(:p\d .br **troff:** \v'-.4m'\s-3\\n+(:p\s0\v'.4m'

HF Heading font list {3.5.2.4.1}
Up to seven codes for heading levels 1 through 7
2 2 2 2 2 2 2 (all levels given emphasis)

HP Heading point size list {3.5.2.5}
Up to seven codes for heading levels 1 through 7

Le Title for LIST OF EQUATIONS {3.9}

Lf Title for LIST OF FIGURES {3.9}

Lt Title for LIST OF TABLES {3.9}

Lx Title for LIST OF EXHIBITS {3.9}

RE SCCS Release and Level of **mm** {6.9}
Release.Level (for example, 15.129)

Rf Reference number generator {5.8}

Rp Title for References {5.8}

Tm Trademark string {1.4.8}
Places the letters "TM" one-half line above the text that it follows Seven accent strings are also available. (See section 6.6.)

9.3. Appendix C: mm Number Register Summary

The list that follows contains a description of all the predefined number registers used by **mm**. Numbers enclosed in braces {} show the section in this technical discussion where more information about the register can be found. After each description is the normal value of the register followed by the range of allowable values, enclosed in brackets []. The lower and upper limit of values are separated by a colon (:). An asterisk attached to a register name means that this register can be set only from the command line or before the **mm** macro definitions are read by the formatter. Section 1.2 has notes on setting and referencing registers. Any register having a single-character name can be set from the command line {8.4}. These are marked with a dagger (†) in the following list.

A *† Handles preprinted forms and logo
{8.4} 0, [0:2]

Au Inhibits printing of author information
{4.1.5} 1, [0:1]

C *† Copy type (original, DRAFT, and so on)
{8.4} 0 (Original), [0:4]

Cl Level of headings saved for table of contents
{3.5.3} 2, [0:7]

Cp Placement of list of figures, and so on
{3.9} 1 (on separate pages), [0:1]

D *† Debug flag
{8.4} 0, [0:1]

De Display eject register for floating dislays
{3.6.2} 0, [0:1]

Df Display format register for floating displays
{3.6.2} 5, [0:5]

Ds Static display pre- and post-space
{3.6.1} 1, [0:1]

E *† Controls font of the subject:/date:/from: fields
{8.4} 1 (**nroff**) 0 (**troff**), [0:1]

Ec Equation counter, used by **.EC** macro
{3.9} 0, [0:?]
Incremented by one for each **.EC** call.

Ej Page-ejection flag for headings
{3.5.2.1} 0 (no eject), [0:7]

Eq Equation label placement
{3.8} 0 (right-adjusted), [0:1]

Ex Exhibit counter, used by **.EX** macro
{3.9} 0, [0:?]
Incremented by one for each **.EX** call.

Fg Figure counter, used by **.FG** macro
{3.9} 0, [0:?]
Incremented by one for each **.FG** call.

Fs Footnote space (that is, spacing between footnotes)
{3.3.2} 1, [0:?]

H1-H7 Heading counters for levels 1 through 7
{3.5.2.6} 0, [0:?]
Incremented by the **.H** macro of corresponding. level or the **.HU** macro if at level given by the **Hu** register. The **H2** through **H7** registers are reset to by any **.H** (**.HU**) macro at a lower-numbered level.

Hb Heading break level (after **.H** and **.HU**)
{3.5.2.2} 2, [0:7]

Hc Heading centering level for **.H** and **.HU**
{3.5.2.3} 0 (no centered headings), [0:7]

Hi Heading temporary indent (after **.H** and **.HU**)
{3.5.2.2} 1 (indent as paragraph), [0:2]

Hs Heading space level (after **.H** and **.HU**)
{3.5.2.2} 2 (space only after **.H 1** and **.H 2**), [0:7]

Ht Heading type (for **.H**: single or concatenated numbers)
{3.5.2.6} 0 (concatenated numbers: 1.1.1, and so on), [0:1]

Hu Heading level for unnumbered heading (**.HU**)
{3.5.1} 2 (**.HU** at the same level as **.H 2**), [0:7]

Hy Hyphenation control for body of document
{1.4.4} 0 (automatic hyphenation off), [0:1]

L *† Length of page
{8.4} 66, [20:?] .br (11i, [2i:?] in **troff** formatter)

Le List of equations
{3.9} 0 (list not produced) [0:1]

Lf List of figures
{3.9} 1 (list produced) [0:1]

Li List indent
{3.2.1} 6 (**nroff**) 5 (**troff**), [0:?]

Ls List spacing between items by level
{3.2.1} 6 (spacing between all levels) [0:6]

Lt List of tables
{3.9} 1 (list produced) [0:1]

Lx List of exhibits
{3.9} 1 (list produced) [0:1]

N *† Numbering style
{8.4} 0, [0:5]

Np Numbering style for paragraphs
{3.1.2} 0 (unnumbered) [0:1]

O *† Offset of page
{8.4} .75i, [0:?] .br (0.5i, [0i:?] in **troff** formatter) .
For **nroff** formatter, these values are unscaled numbers representing lines or character positions. For **troff** formatter, these values must be scaled.

Oc Table of contents page numbering style
{5.4.1} 0 (lower-case roman), [0:1]

Of Figure caption style
{3.9} 0 (period separator), [0:1] P † Page number managed by **mm**
{8.4} 0, [0:?]

Pi Paragraph indent
{3.1.3} 5 (**nroff**) 3 (**troff**), [0:?]

Ps Paragraph spacing
{3.1.1} 1 (one blank space between paragraphs), [0:?]

Pt Paragraph type
{3.1.1} 0 (paragraphs always left justified), [0:2]

Pv "PRIVATE" header
{3.4.9} 0 (not printed), [0:2]

Rf Reference counter, used by **.RS** macro
{5.8} 0, [0:?]
Incremented by one for each **.RS** call. .LI **S** * † The **troff** formatter default point size
{8.4} 10, [6:36]

Si Standard indent for displays
{3.6.1} 5 (**nroff**) 3 (**troff**), [0:?]

T * † Type of **nroff** output device
{8.4} 0, [0:2]

Tb Table counter, used by **.TB** macro
{3.9} 0, [0:?]
Incremented by one for each **.TB** call.

U * † Underlining style (**nroff**) for **.H** and **.HU**
{8.4} 0 (continuous underline when possible), [0:1]

W * † Width of page (line and title length)
{8.4} 6i, [10:1365] .br (6i, [2i:7.54i] in the **troff** formatter)

9.4. Appendix D: ERROR MESSAGES

While processing, **mm** will report a variety of errors in the usage of both macros and basic formatter primitives. An **mm** error message has a standard part followed by a variable part. The standard part has the form:

ERROR:(filename) input line *n*:

The variable part consists of a descriptive message, usually beginning with a macro name. The variable parts are listed alphabetically by macro name below:

CS:cover sheet too long

Text of the cover sheet is too long to fit on one page. The abstract should be reduced or the indent of the abstract should be decreased.

DE:no DS or DF active

A **.DE** macro has been encountered, but there has not been a previous **.DS** or **.DF** macro to match it.

DF:illegal inside TL or AS

Displays are not allowed in the title or abstract.

DF:missing DE

A **.DF** macro occurs within a display, that is, a **.DE** macro has been omitted or mistyped.

DF:missing FE

A display starts inside a footnote. The likely cause is the omission (or misspelling) of a **.FE** macro to end a previous footnote.

DF:too many displays

More than 26 floating displays are active at once, that is, have been accumulated but not yet output.

DS:illegal inside TL or AS

Displays are not allowed in the title or abstract.

DS:missing DE

A **.DS** macro occurs within a display, that is, a **.DE** has been omitted or mistyped.

DS:missing FE

A display starts inside a footnote. The likely cause is the omission (or misspelling) of a **.FE** to end a previous footnote.

FE:no FS active

A **.FE** macro has been encountered with no previous **.FS** to match it.

FS:missing DE

A footnote starts inside a display, that is, a **.DS** or **.DF** occurs without a matching **.DE**.

FS:missing FE

A previous **.FS** macro was not matched by a closing **.FE**, that is, an attempt is being made to begin a footnote inside another one.

H:bad arg:*value*

The first argument to the **.H** macro must be a single digit from one to seven, but *value* has been supplied instead.

H:missing arg

The **.H** macro needs at least one argument.

H:missing DE

A heading macro (**.H** or **.HU**) occurs inside a display.

H:missing FE

A heading macro (**.H** or **.HU**) occurs inside a footnote.

HU:missing arg

The **.HU** macro needs one argument.

LB:missing arg(s)

The **.LB** macro requires at least four arguments.

LB:too many nested lists

Another list was started when there were already six active lists.

LE:mismatched

The **.LE** macro has occurred without a previous **.LB** or other list- initialization macro {3.2}. This is not a fatal error. The message is issued because there exists some problem in the preceding text.

LI:no lists active

The **.LI** macro occurred without a preceding list-initialization macro. The latter probably has been omitted or entered incorrectly.

LO: Required argument missing

The **.LO** macro requires an argument.

LO: LO argument not recognized

You have provided an argument to **.LO** that it does not recognize.

LT: LT argument not recognized

You have provided an argument to **.LT** that it does not recognize.

ML:missing arg

The **.ML** macro requires at least one argument.

ND:missing arg

The **.ND** macro requires one argument.

RF:no RS active

The **.RF** macro has been encountered with no previous **.RS** to match it.

RP:missing RF

A previous **.RS** macro was not matched by a closing **.RF**.

RS:missing RF

A previous **.RS** macro was not matched by a closing **.RF**.

S:bad arg:value

The incorrect argument *value* has been given for the **.S** macro {6.5}.

SA:bad arg:value

The argument to the **.SA** macro (if any) must be either 0 or 1. The incorrect argument is shown as *value*.

SG:missing DE

The **.SG** macro occurred inside a display.

SG:missing FE

The **.SG** macro occurred inside a footnote.

SG:no authors

The **.SG** macro occurred without any previous **.AU** macro(s).

Check TL, AU, AS, AE, MT sequence

The correct order of macros at the start of a memorandum is shown in section 4.1.8. Something has disturbed this order.

Check TL, AU, AS, AE, NS, NE, MT sequence

The correct order of macros at the start of a memorandum is shown in section 4.1.8. Something has disturbed this order. (Occurs if the **.AS 2** {4.1.9} macro was used.)

VL:missing arg

The **.VL** macro requires at least one argument.

Check WA, WE, IA, IE, LT sequence

The correct order of these macros is shown in section 4.2.6. Something has disturbed this order.

)W: WA macro missing

If you use **.LT**, you must specify at least one **.WA**-**.WE** pair.

)W: WA or WE macro missing

If you use **.WA** or **.WE**, you must specify the other member of the macro pair.

)W: WA, WE, or IE macro missing

You have omitted either or both of the **.IA** and **.IE** macros.

WC:unknown option

An incorrect argument has been given to the **.WC** macro.

9.5. Appendix E: The Define File— /usr/lib/macros/strings.mm

```
.\"                Copyright (c) 1984 AT&T
.\"                  All Rights Reserved
.\"     THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF AT&T
.\"   The copyright notice above does not evidence any actual
.\"   or intended publication of such source code.
.\"        @(#)macros:strings.mm        1.5
.\"        .\"tab  begins comments.
.\"        No comments should appear on the same line as the string
.\"        definition.
.\"
.\"        The following string is used by the macro MT.
.\"        ]S defined as logo character
.ds ]S
.\"        ]Z Company Name
.ds ]Z AT&T Bell Laboratories
.\"
.\"        The following strings are used by the macro PM
.\"
.\"        Proprietary marking "PM1" (BP, N, P and BPN) lines 1-4
.ds ]M AT&T BELL LABORATORIES - PROPRIETARY
.ds ]O Use pursuant to G.E.I. 2.2
.ds ]Q
.ds ]R
.\"        Proprietary marking "PM2" (CA) lines 1-4
.ds ]A THIS DOCUMENT CONTAINS PROPRIETARY INFORMATION OF
.ds ]F AT&T AND IS NOT TO BE DISCLOSED OR USED EXCEPT IN
.ds ]G ACCORDANCE WITH APPLICABLE CONTRACTS OR AGREEMENTS.
.ds ]H
.\"        Proprietary marking "PM3" (CP) lines 1-4
.ds ]I SEE PROPRIETARY NOTICE ON COVER PAGE
.ds ]J
.ds ]K
.ds ]L
.\"        Proprietary marking "PM4" (BPP and BR) lines 1-4
.ds ]U AT&T BELL LABORATORIES - PROPRIETARY (RESTRICTED)
.ds ]V Solely for authorized persons having a need to know
```

```
.ds ]W pursuant to G.E.I. 2.2
.ds ]X
.\"         Proprietary marking "PM5" (ILL) lines 1-4
.ds ]i THIS DOCUMENT CONTAINS PROPRIETARY INFORMATION OF
.ds ]j AT&T BELL LABORATORIES AND IS NOT TO BE DISCLOSED,
.ds ]k REPRODUCED, OR PUBLISHED WITHOUT WRITTEN CONSENT.
.ds ]l THIS DOCUMENT MUST BE RENDERED ILLEGIBLE WHEN BEING DISCARDED.
.\"         Proprietary marking "PM6" (CI-II) lines 1-4
.ds ]m CI-II
.ds ]o Not for disclosure to AT&T Informations Systems.
.ds ]p Subject to FCC separation requirements under Computer Inquiry II
.ds ]q
```

9.6. Appendix F: Sample Footnotes

Input:

```
.FD 10
.P
This example illustrates several footnote styles
for both labeled and automatically numbered footnotes.
First input, then output is shown.
With the footnote style set to the
default style,
the first footnote is processed\*F
.FS
This is the first footnote text example.
This is the default style (.FD 10).
The right margin is not justified,
hyphenation is not permitted,
text is indented, and the automatically generated label is
right justified in the text-indent space.
.FE
and followed by a second footnote.*****
.FS *****
This is the second footnote text example.
This is also the default style (.FD 10)
but with a long footnote label (*****) provided by the user.
.FE
.FD 1
Footnote style is changed by using the .FD macro to
specify hyphenation, right margin justification,
indentation, and left justification of the label.
This produces the third footnote,\*F
.FS
This is the third footnote example (.FD 1).
The right margin is justified, the footnote text is indented,
and the label is left justified in the text-indent space.
Although not necessarily illustrated by this example,
hyphenation is permitted.
.FE
and then the fourth footnote.†
.FS †
```

```
This is the fourth footnote example (.FD 1).
The style is the same as the third footnote.
.FE
.FD 6
Footnote style is set again via the .FD macro for no hyphenation,
no right margin justification,
no indentation, and with the label left justified.
This produces the fifth footnote.\*F
.FS
This is the fifth footnote example (.FD 6).
The right margin is not justified, hyphenation is not permitted,
footnote text is not indented,
and the label is placed at the beginning of the first line.
.FE
```

Output:

```
                              - 1 -
This example illustrates several footnote styles for both
labeled and automatically numbered footnotes.  First input,
then output is shown.  With the footnote style set to the
default style, the first footnote is processedl and followed
by a second footnote.***** Footnote style is changed by
using the .FD macro to specify hyphenation, right margin
justification, indentation, and left justification of the
label.  This produces the third footnote,2 and then the
fourth footnote.┼ Footnote style is set again via the .FD
macro for no hyphenation, no right margin justification, no
indentation, and with the label left justified.  This
produces the fifth footnote.3

_____

 1. This is the first footnote text example.  This is the
    default style (.FD 10).  The right margin is not
    justified, hyphenation is not permitted, text is
    indented, and the automatically generated label is right
    justified in the text-indent space.
***** This is the second footnote text example.  This is
    also the default style (.FD 10) but with a long footnote
    label (*****) provided by the user.
2.  This  is  the third footnote example (.FD 1).  The right
    margin is justified, the footnote text is indented,  and
    the  label  is  left justified in the text-indent space.
    Although not necessarily illustrated  by  this  example,
    hyphenation is permitted.
┼   This  is the fourth footnote example (.FD 1).  The style
    is the same as the third footnote.
3. This is the fifth footnote example (.FD 6).  The right
margin is not justified, hyphenation is not permitted,
footnote text is not indented, and the label is placed at
the beginning of the first line.
```

9.7. Appendix G: Formal Memorandum

Input:

```
.ND "September 28, 1984"
.TL
Document Production Coordinator
.AU "John Smith" JJS XF 5414 6398 7-123 machine_5!jjs
.AF "Business Computer Systems, Inc."
.MT 0
.P
Business Computer Systems, Inc. (BCS) has job
openings for people that
can do the following tasks:
.AL
.LI
Develop methods for producing
documentation
.LI
Perform duties resulting from the development of these methods.
For example,
.BL
.LI
Use text processing with Documenter's Toolkit
.FS *
Trademark of Data General Corp.
.FE
(mm, nroff, tbl) to:
.DL
.LI
Prepare documents and tables
.LI
Develop new macro commands
.LE
```

```
.LI
Serve as a point of contact for BCS with printers and
distributors.
.LE
.LI
If the job holder's interests and writing skills match the
needs of the Technical Writing Staff,
write documents.
.LE
.SG
.NS
BCS Technical Writing Staff
.NE
```

Output:

AT&T Business Computer Systems, Inc.

subject: Document Production Coordinator

date: September 28, 1984

from: John Smith
XF 5414
7-123 x6398
machine_5!jjs

Business Computer Systems, Inc. (BCS) has job openings for people that can do the following tasks:

1. Develop methods for producing documentation

2. Perform duties resulting from the development of these methods. For example:

   - Use text processing with Documenter's Toolkit* (mm, nroff, tbl) to:

     - Prepare documents and tables

     - Develop new macro commands

   - Serve as a point of contact for BCS with printers and distributors.

3. If the job holder's interests and writing skills match the needs of the Technical Writing Staff, write documents.

John Smith

Copy to
BCS Technical Writing Staff

---

* Trademark of Data General Corp.

9.8. Appendix H: Business Letter

Input:

```
.LO AT "Personnel"
.WA "Bill Taylor" "Director of Placement Services"
Columbia University
116th Street
New York, NY 10019
.WE
.LO SA "Dear Dr. Smith:"
.LO RN "Career Day"
.IA "Rebecca Smith"
Business Computer Systems, Inc.
190 River Boulevard
Durham, N.C. 27707
.IE
.LT BL
.P
We would be happy to include a representative of your company in our
"Career Day" program this spring.
I am forwarding your request to James Blinn, who is organizing the event
this year.
.P
Thank you for your interest in our placement programs.
.FC "Sincerely,"
.SG
.NS 5
.NE
.NS
J. Blinn
.NE
```

Output:

```
                                - 1 -


                                Columbia University
                                116th Street
                                New York, NY 10019
                                December 15, 1985

                                In reference to:  Career Day


Rebecca Smith
Business Computer Systems, Inc.
190 River Boulevard
Durham, N.C.  27707

ATTENTION:  Personnel

Dear Dr. Smith:

We would be happy to include a representative of your
company in our "Career Day" program this spring. I am
forwarding your request to James Blinn, who is organizing
the event this year.

Thank you for your interest in our placement programs.


                                Sincerely,



                                Bill Taylor
                                Director of Placement Services

JL-der
Enc.
Copy to
J. Blinn
```

tbl: Technical Discussion

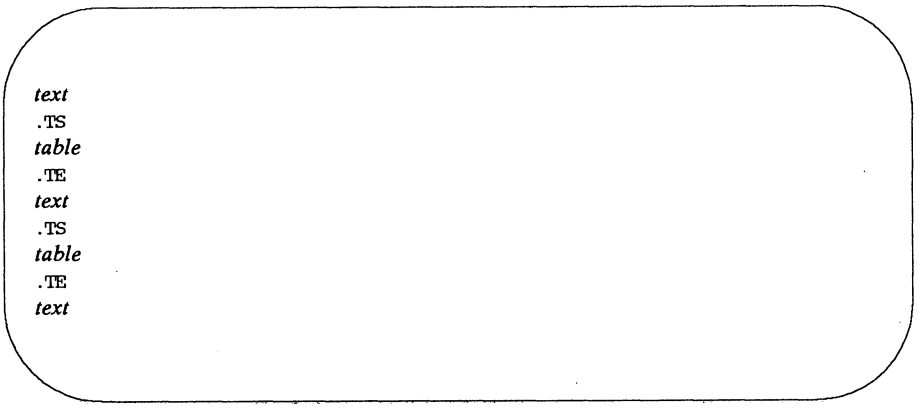
1. Introduction

This is a technical discussion of **tbl**, a program that you use to produce simple and complex tables. Read this if you have had some experience with **tbl**; if you have never used **tbl** before, you should read the tutorial "The Preprocessor **tbl**" in *Using the Documenter's Tool Kit on the DG/UX System*.

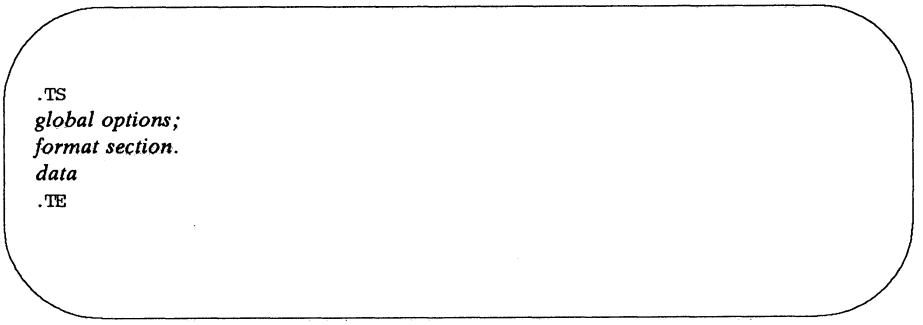
Tables that you format with **tbl** consist of columns that you may center, right-adjust, left-adjust, or align according to digit and decimal point. You may place labels over single columns or groups of columns. A table entry may contain equations or consist of several rows of text. You may draw horizontal or vertical lines in the table, and you may enclose any table or element in a box.

2. Table Delimiters (.TS and .TE)

Delimit tables with the macro pair **.TS** (table start) and **.TE** (table end). **tbl** processes certain commands (options, key letters, and commands specifying such details as point size and font control) that it finds between this macro pair, and it generates formatting requests, escape sequences, defined strings, and number registers. The **.TS** and **.TE** lines are copied so that **nroff/troff** formatter layout macros (such as **mm** formatting macros) can use these lines as delimiters. The general format of a file that you would submit to **tbl** is



The format of each table is



Each table is independent and contains

- Global options that affect the layout of the whole table
- A format section that describes individual columns and rows of the table
- Data that you want printed

Unlike the options section, the format section and data are always required.

3. Global Options

You may specify a single line of options to affect the layout of the whole table. These must be placed immediately after the **.TS** line. On this line, you must separate options with spaces, tabs, or commas. You must end the options line with a semicolon. Allowable options are

center	centers the table (default is left-adjusted)
expand	makes the table as wide as current line length
box	encloses the table in a box
allbox	encloses each data entry of the table in a box
doublebox	encloses the table in two boxes
tab (*x*)	separates data entries by using *x* instead of tab
linesize (*n*)	sets lines or rules (for example, from **box**) in *n*-point type
delim (*xx*)	recognizes *x* and *x* as **eqn** delimiters.

4. Format Section

The format section of the table specifies the layout of the columns. Each line in the format section corresponds to one line of table data except the last format line, which corresponds to all following data lines up to any additional **.T&** command line. (**.T&** is described below.) Each format line contains a key letter for each column of the table.

4.1. Key Letters

In the format section, you may separate key letters with spaces or tabs to improve readability, but spaces or tabs are not necessary. A dot (**.**) indicates the end of key letters and should follow the last key letter before data is entered in the lines below. Key letters are

L or **l** specifies a left-adjusted column entry

R or **r** specifies a right-adjusted column entry

C or **c** specifies a centered column entry

N or **n** specifies a numerical column entry. Numerical entries are aligned according to digit and decimal point.

A or **a** specifies an alphabetic subcolumn. All entries in an alphabetic subcolumn are aligned on the left and positioned so that the widest entry is centered within the column.

S or **s** specifies a spanned heading. The entry from the previous column continues across this column. You may not use this key letter for the first column of a table.

^ specifies a vertically spanned heading. The entry from the previous row continues down through this row You may not use this key letter for the first row of a table.

The layout of key letters in the format section resembles the layout of the data in the table. Thus, a simple 3-column format might appear as

```
css
lnn.
```

The first line of the table contains a heading centered across all three columns. Each remaining line of the table contains a left-adjusted item in the first column followed by two columns of numerical data. A sample table in this format is

CENTERED HEADING		
Item-a	34.22	9.1
Item-b	12.65	.02
Item-c	23	5.8
Total	69.87	14.92

Instead of listing the format of successive lines of a table on consecutive lines of the format section, you may give successive line formats on the same line, separated by commas. That is, you can specify the format for the above example like this:

```
c s s, l n n.
```

Again, signal the end of the key letter section with a dot (.).

When you specify numerical column alignment (*n*), **tbl** seeks a location for the decimal point. The rightmost dot (.) adjacent to a digit is used as a decimal point. If there is no dot adjoining a digit, the rightmost digit is used as a units digit. If you do not specify alignment, the item is centered in the column. However, you may use the escape sequence **\&** (a zero-width character) to override dots and digits or to align alphabetic data. This aligns the dots, and the **\&** disappears from the final output. In the following example, the output column shows how items in the input column are aligned in a numerical column.

```
input:                  output:

.TS
center;
n.
13                        13
4.2                        4.2
26.4.12                  26.4.12
abcdefg                 abcdefg
abcd\&efg                abcdefg
abcdefg\&               abcdefg
43\&3.22                      433.22
749.12                   749.12
.TE
```

If you use numerical data in the same column with wider **L** or **r** type table entries, **tbl** centers the widest number relative to the wider **L** or **r** items and preserves alignment within the numerical items. This is similar to the behavior of **a** type data. Alphabetic subcolumns (requested by the **a** key letter) are always slightly indented relative to **L** items. If necessary, the column width is increased to force this. This is not true for **n** type entries. You should not use **n** and **a** items in the same column.

4.2. Additional Features

Additional features of the key letter system are

Horizontal lines

You may replace a key letter by an underscore (_) to specify a horizontal line in place of the column entry, or an equal sign (=) to specify a double horizontal line. If an adjacent column contains a horizontal line or if there are vertical lines adjoining this column, the horizontal line extends to meet nearby lines. If you provide any data entry for a column containing an underscore or an equals sign, **tbl** ignores the entry and prints a warning message.

Vertical lines

If you place a vertical bar (|) between column key letters, a vertical line appears between the corresponding columns of the table. A vertical bar to the left of the first key letter or to the right of the last one

produces a line at the edge of the table. If two vertical bars appear between key letters, **tbl** draws a double vertical line.

Space between columns

You may follow a key letter with a number to show the amount of separation between this column and the next. The number specifies the separation in ens. One en is a relative measure about the width of the letter "n" in the current point size. More precisely, an en is the number of points (1 point = 1/72 inch) equal to half the current type size. If you use the **expand** global option, these numbers are multiplied by a constant so that the table is as wide as the current line length. The default column separation number is 3. If you change the separation number, the largest space requested prevails.

Vertical spanning

Vertically spanned entries extending over several rows of the table are centered in their vertical range. If you follow a key letter with **t** or **T**, any corresponding vertically spanned item begins at the top line of its range.

Font changes

You may specify that the corresponding column should be in a different font from the default font (usually roman) by following a key letter with the letter **f** or **F** and a character naming the desired font. You should put a space or a tab between a 1-letter font name and whatever follows. The single letters **B**, **b**, **I**, and **i** are shorter synonyms for **fB** and **fI**. Font-change requests given with the data override these specifications.

Point size changes

You may specify the point size of the corresponding table entries by following a key letter with **p** or **P** and a number. If **p** or **P** is followed by an arithmetic expression, such as **+2**, the formatter interprets it as an increment (or decrement in the case of **−2**) from the current point size. If you give both a point size and a column separation value, you must separate them with one or more blanks.

Vertical spacing changes

You may specify the vertical line spacing used within a multi-line data entry by following a key letter with **v** or **V** and a number. The formatter interprets an arithmetic expression (**+2** or **−2**, for example) as an increment or decrement from the current vertical spacing. You must separate a column separation value by blanks or some other specification from a vertical spacing request. This request has no

effect unless the corresponding data entry is a text block.

Column width indication
: You may specify minimum column width by following a key letter with **w** or **W** and a width value in parentheses (for example, **cw(5)**). If the largest element in the column is not as wide as this specified width value, the formatter uses the width value to determine column size. If the largest element in the column is wider than the value you specify, the width of the largest element determines column width.

  The formatter also uses the width value as a default line length for included text blocks. You can use normal **nroff/troff** formatter dimensions, such as **i**, **m**, **n**, **v**, or **u** to scale the width value. (See the "**nroff/troff** Technical Discussion".) If you do not specify a scale, the formatter uses ens. If the width value is an unscaled integer, you may omit the parentheses. If you give another width value in a column, the last one you specify controls the width.

Equal-width columns
: If you follow a key letter by **e** or **E**, you specify equal-width columns. The formatter makes all columns whose key letters are followed by **e** or **E** the same width.

Staggered columns
: You may specify that the corresponding entry is to be moved up one-half line by following a key letter with **u** or **U**. This makes it easy to have a column of differences between numbers in an adjoining column. The **allbox** option does not work with staggered columns.

Zero-width item
: A key letter followed by **z** or **Z** specifies that the corresponding data item is to be ignored in calculating column widths. This may be useful in allowing headings to run across adjacent columns where spanned headings would be inappropriate.

Default
: Column descriptors missing from the end of a format line are assumed to be **L**. That is, if you have more columns of data than descriptors to specify them, the data in the unspecified columns are left-justified. The longest line in the format section defines the number of columns in the table. **tbl** ignores columns in the data if there are not corresponding key letters in the format section.

As some examples in the "**tbl** Sampler," the features need not be separated by spaces except to avoid ambiguities involving point size and font changes. Thus, a numerical column entry in italic font and 12-point type with a minimum width of 2.5 inches and separated by 6 ens from the next column could be specified as

```
np12w(2.5i)fI 6
```

5. Data

Put table data after the format section, typing each line of the table as one line of data. You may break a long input line into smaller lines by ending each short line, except the last, with a backslash. The resulting output line is as long as the combined parts.

Data to be placed in different columns (data entries) should be separated with tabs or with whatever character you specify in the **tab** global option. There are a few special cases of data entries:

nroff/troff commands within tables
: **tbl** assumes that an input line beginning with a dot is a request to the formatter and interprets it accordingly, retaining its position in the table. For example, a space within a table may be produced with the **.sp** request in the data.

Full width horizontal lines
: **tbl** interprets a data line containing only the _ (underscore) character or = (equal sign) to be a single or double line, respectively, extending the full width of the table.

Single column horizontal lines
: **tbl** interprets a data entry containing only the _ character or the = to be a single or double line extending the full width of the column. Such lines extend to meet horizontal or vertical lines adjoining this column. To obtain these characters explicitly in a column, precede them with the escape sequence \&, or follow them with a space before the usual tab or newline character.

Short horizontal lines
: A data entry containing only the string _ is assumed to be a single line as wide as the contents of the column. It does not extend to meet adjoining lines.

Repeated characters
: A data entry containing only a string of the form \R*x*, where *x* is any character, is replaced by repetitions of the character *x* as wide as data in the column. The sequence does not extend to meet adjoining columns.

Vertically spanned entries
: A data entry containing only the escape sequence \^ specifies that the data entry immediately above it spans downward over the corresponding row. It is equivalent to a table format key letter of ^.

Text blocks

To include a block of text as a data entry, precede it by **T{** and follow it by **T}**. Thus, the sequence

```
. . . T{
block of
text
T} . . .
```

is a convenient method for inserting data not easily placed between tabs. You may put **T{** (the beginning delimiter) anywhere in the data section. Do not put any character to the immediate right of this delimiter. You must put **T}** (the end delimiter) at the beginning of a line, and you may follow **T}** with a tab or tab character. You may follow the tab with additional columns of data on the same line. Table 4 in the "**tbl** Sampler" shows how to use text blocks.

Text blocks are pulled out from the table, processed separately by the formatter, and replaced in the table as a solid block. Various limits in the **nroff/troff** program are likely to be exceeded if 30 or more text blocks are used in a table. This produces diagnostic messages such as `Too many text block diversions; tbl quits`, `Too many string/macro names`, or `Too many number registers`.

If no line length is specified in the block of text or in the table format, the default is to use

$$L \times C / (N + 1)$$

where L is the current line length, C is the number of table columns spanned by the text, and N is the total columns in the table.

Other parameters (point size, font, and so on) that you may use in typesetting the text block are

- those in effect at the beginning of the table (including the effect of the **.TS** macro)
- any table format specifications of size, spacing, and font using the **p**, **v**, and **f** modifiers to the column key letters
- **nroff/troff** requests within the text block itself (requests within the data but not within the text block itself do not affect that block).

Although you may put many lines in a table, **tbl** uses only the first 200 lines to create the table. Similarly, column adjustment is determined for the first 200 lines only. You may arrange a multi-page table as several single-page tables if these limits prove to be a problem.

When calculating column widths, **tbl** assumes that all data entries are in the font and size being used when the formatter encounters **.TS**. This is true except for font and size changes specified in the table format section (see Table 7 in the "**tbl** Sampler" or within the table data (as mentioned in the section "Data"). You may sprinkle **nroff/troff** requests throughout a table, but take care not to confuse **tbl**'s width calculations.

6. Additional Command Lines

To change the format of a single table, use the **.T&** (table continue) command. Such a table would look like this:

```
.TS
global options ;
format section .
data
. . .
.T&
new format section .
data
.T&
newer format section .
data
.TE
```

Using this procedure, each table line can be close to its corresponding format line. It is not possible to change the number of columns, the space between columns, the global options such as **box**, or the selection of columns to be made equal in width in additional command lines. **tbl** does not recognize this command after the first 200 lines of a table.

7. Using the tbl Command

As the term "preprocessor" implies, you process a file with **tbl** before you run **nroff** or **troff**. On the DG/UX system, the **tbl** program can process a simple table with the **nroff** commands

tbl **−TX** *file* | **nroff** [*option(s)*]

or

mm **−t** [*option(s)*] *file*

For **troff** output you can use

tbl *file* | **troff** [*option(s)*]

or

mmt **−t** [**option(s)**] *file*

When there are several input files containing tables, equations, pictures, and **mm** macro requests, type

tbl **−TX** *file1 file1...* | **neqn** | **nroff** **−mm**

For **troff** output type

tbl **−TX** *file1 file2...* | **eqn** | **troff** **−mm**

The next section explains the rationale for placing **tbl** before **eqn**).

Notice that you may specify options usually used with the **nroff** formatter. Using **nroff/troff** is similar to using **nroff**. If you specify " **−** " as the file name, **tbl** reads the standard input at that point.

The special **-TX** command-line option to **tbl** produces output without fractional line motions to accommodate line printers without adequate driving tables or postprocessors. That is, this option tells **tbl** to process data in terms of full line motions.

7.1. Equations in Tables

When both **tbl** and **eqn** programs operate on the same file, you should call **tbl** first. If there are no equations within tables, either sequence works. It is usually faster to execute **tbl** first because **eqn** normally produces a larger expansion of the input. However, if you have equations within tables (using **eqn**'s **delim** statement, see Table 5 in the "**tbl** Sampler"), you must execute **tbl** first, or the output will be scrambled. You should avoid use of **eqn**'s equations in *n*-style (numeric) columns since **tbl** attempts to split numerical format items into two parts, and the **delim** (*xx*) statement prevents splitting numerical columns within delimiters. For example, if the **eqn** delimiters are **$$**, a **delim** (**$$**) statement causes a numerical column entry such as

```
1245 $\(+- 16$
```

to be divided after 1245, not after 16.

7.2. Using Number Registers

The **tbl** program accepts up to 35 columns of data; this figure may be compromised depending on the availability of **nroff/troff** formatter number registers. **tbl** itself must define several macros and number registers. As a result, you must not use the following names for macros that you create: **#f, #o, #+, #%, #&,** and **#**. You must also avoid using number registers named by **tbl**. These include 2-digit numbers from 31 to 99 and strings of the form **4***x*, **5***x*, **#***x*, *x*+, *x*|, ^*x*, and *x*−, where *x* is any lower-case letter. The register names **##**, **#−**, **#^**, **#T**, and **T#** are also used by **tbl** in certain circumstances. To conserve register names, the **n** and **a** key letters share a register; hence the restriction that you may not use them in the same column of the format section.

As an aid in writing layout macros, **tbl** defines the number register **TW**, which is the table width, by the time that the **.TE** macro is invoked. You may use the register in the expansion of **.TE**. More important, to assist in laying out multi-page boxed tables, the macro **T#** is defined to produce the bottom lines and side lines of a boxed table and then to be invoked at its end.

You can print a multi-page boxed table with a repeated heading by giving the argument **H** to the **.TS** macro. If the table start macro is written

```
.TS H
```

then, a line of the form

```
.TH
```

must be given in the table after any table heading (or at the start if none). Material up to the **.TH** is placed at the top of each page of the table. The remaining lines in the table are placed on several pages as required. This is a feature of the **mm** macros, not of **tbl**.

8. tbl Sampler

The following tables illustrate basic concepts of the **tbl** program. The <TAB> symbol in the input data represents a tab character produced by typing **\t** or by pressing the tab key. Although each table shows particular options or features, you may glean other information about usage from them.

The following pages contain successive examples of matching input and output to provide models for making a variety of tables. The general format of the "**tbl** Sampler" is to show the contents of an input file, as read at a terminal, on the left page:

Input:

```
.TS
option option option option;
key letter  descriptor  key letter  descriptor
key letter  descriptor  key letter  descriptor
key letter  descriptor  key letter  descriptor
key letter  descriptor  key letter  descriptor.
table heading data
optional double or single horizontal line
data entry <TAB> data entry <TAB> data entry
optional double or single horizontal line
data entry <TAB> data entry <TAB> data entry
optional double or single horizontal line
data entry <TAB> data entry <TAB> data entry
.TE
```

On the opposite right page, a corresponding example of output appears. **Output:**

table heading data		
data entry	*data entry*	*data entry*
data entry	*data entry*	*data entry*
data entry	*data entry*	*data entry*

The command line used for most of the examples below is as follows:

tbl *file* | **troff** | *phototypesetter*

Table 5 required

tbl *file* | **eqn** | **troff** | *phototypesetter*

in order to process the equations made with **eqn**.

Table 1: Using Horizontal Lines in Place of Key Letters

Input:

```
.TS
box;
L L L
L L _
L L | LB
L L _
L L L.
january <TAB> february <TAB> march
april <TAB> may
june <TAB> july <TAB> Months
august <TAB> september
october <TAB> november <TAB> december
.TE
```

Output:

january	february	march
april	may	**Months**
june	july	
august	september	
october	november	december

Table 2: Changing Point Size of One Column

Input:

```
.TS
c c
np-2 | n | .
<TAB>Stack
<TAB>_
1<TAB>46
<TAB>_
2<TAB>23
<TAB>_
3<TAB>15
<TAB>_
4<TAB>6.5
<TAB>_
5<TAB>2.1
<TAB>_
.TE
```

Output:

	Stack
1	46
2	23
3	15
4	6.5
5	2.1

Table 3: Using Additional Command Lines

Input:

```
.TS
box,center;
cf3 s s s.
Composition of Foods
_
.T&
c | c s s
c | c s s
c | c | c | c.
Food <TAB> Percent by Weight
\^ <TAB> _
\^ <TAB> Protein <TAB> Fat <TAB> Carbo-
\^ <TAB> \^ <TAB> \^ <TAB> hydrate
_
.T&
l | n | n | n.
Apples <TAB> .4 <TAB> .5 <TAB> 13.0
Halibut <TAB> 18.4 <TAB> 5.2 <TAB> . . .
Lima beans <TAB> 7.5 <TAB> .8 <TAB> 22.0
Milk <TAB> 3.3 <TAB> 4.0 <TAB> 5.0
Mushrooms <TAB> 3.5 <TAB> .4 <TAB> 6.0
Rye bread <TAB> 9.0 <TAB> .6 <TAB> 52.7
.TE
```

Output:

Composition of Foods			
Food	Percent by Weight		
	Protein	Fat	Carbo-hydrate
Apples	.4	.5	13.0
Halibut	18.4	5.2	...
Lima beans	7.5	.8	22.0
Milk	3.3	4.0	5.0
Mushrooms	3.5	.4	6.0
Rye bread	9.0	.6	52.7

Table 4: Using Text Blocks and Controlling Column Width

Input:

```
.TS
allbox;
cf2 s s
cw(1i) cw(1.5i) cw(1.5i)
l l l.
New York Area Rocks
.sp
Era <TAB> Formation <TAB> Age (years)
Precambrian <TAB> Reading Prong <TAB> >1 billion
Paleozoic <TAB> Manhattan Prong <TAB> 400 million
Mesozoic <TAB> T{
Newark Basin, incl.
Stockton,Lockatong,
and Brunswick
formations
T} <TAB> 200 million
Cenozoic <TAB> Coastal Plain <TAB> T{
On Long Island
30,000 years;
Cretaceous sediments
redeposited
by recent
glaciation
T}
.TE
```

Output:

New York Area Rocks		
Era	Formation	Age (years)
Precambrian	Reading Prong	>1 billion
Paleozoic	Manhattan Prong	400 million
Mesozoic	Newark Basin, incl. Stockton, Lockatong, and Brunswick formations	200 million
Cenozoic	Coastal Plain	On Long Island 30,000 years; Cretaceous sediments redeposited by recent glaciation

Table 5: Using Additional Command Lines and Defining Tab Character

Input:

```
.TS
tab( <TAB> );
c s
ci s
l n
a n.
Some London Transport Statistics
(Year 1964)
Railway route miles <TAB> 244
Tube <TAB> 66
Sub-surface <TAB> 22
Surface <TAB> 156
.sp .5
.T&
l r
a r.
Passenger traffic \- railway
Journeys <TAB> 674 million
Average length <TAB> 4.55 miles
Passenger miles <TAB> 3,066 million
.T&
l r
a r.
Passenger traffic \- road
Journeys <TAB> 2,252 million
Average length <TAB> 2.26 miles
Passenger miles <TAB> 5,094 million
.T&
l n
a n.
.sp .5
Vehicles <TAB> 12,521
Railway motor cars <TAB> 2,905
Railway trailer cars <TAB> 1,269
Total railway <TAB> 4,174
Omnibuses <TAB> 8,347
```

Output:

Some London Transport Statistics
(Year 1964)

Railway route miles	244
Tube	66
Sub-surface	22
Surface	156
Passenger traffic – railway	
Journeys	674 million
Average length	4.55 miles
Passenger miles	3,066 million
Passenger traffic – road	
Journeys	2,252 million
Average length	2.26 miles
Passenger miles	5,094 million
Vehicles	12,521
Railway motor cars	2,905
Railway trailer cars	1,269
Total railway	4,174
Omnibuses	8,347

Table 6: Using Different Point Sizes and Column Widths

Input:

```
.TS
box tab(<TAB>);
cbsss
c|c|cs
ltiw(1i)|ltw(1.5i)p8|lp8|lw(1.5i)p8.
Some Interesting Places
_
Name<TAB>Description<TAB>Practical Information
_
T{
American Museum
of Natural History
T}<TAB>T{
The collections fill 11.5 acres
(Michelin) or 25 acres (MTA)
of exhibition halls on
four floors.  There is a full-sized
replica of a blue whale and the
world's largest star sapphire
(stolen in 1964).
T}<TAB>Hours<TAB>10-5, ex. Sun 11-5, Wed. to 9
\^<TAB>\^<TAB>Location<TAB>T{
Central Park West & 79th St.
T}
\^<TAB>\^<TAB>Admission<TAB>Donation: $1.00 asked
\^<TAB>\^<TAB>Subway<TAB>AA to 81st St.
\^<TAB>\^<TAB>Telephone<TAB>212-873-4225
_
Bronx Zoo<TAB>T{
About a mile long and .6 mile
wide, this is the largest zoo in
America.  A lion eats 18
pounds of meat a day while a
sea lion eats 15 pounds of fish.
T}<TAB>Hours<TAB>T{
```

Input Cont'd:

```
10-4:30 winter, to 5:00 summer
T}
\^<TAB>\^<TAB>Location<TAB>T{
185th St. & Southern Blvd, the Bronx.
T}
\^<TAB>\^<TAB>Admission<TAB>$1.00, but Tu,We,Th free
\^<TAB>\^<TAB>Subway<TAB>2, 5 to East Tremont Ave.
\^<TAB>\^<TAB>Telephone<TAB>212-933-1759
_
Brooklyn Museum<TAB>T{
Five floors of galleries contain
American and ancient art.
There are American period
rooms and architectural ornaments
saved from wreckers,
such as a classical figure from
Pennsylvania
Station.
T}<TAB>Hours<TAB>Wed-Sat, 10-5, Sun 12-5
\^<TAB>\^<TAB>Location<TAB>T{
Eastern Pkway & Washington Ave.
Brooklyn.
T}
\^<TAB>\^<TAB>Admission<TAB>Free
\^<TAB>\^<TAB>Subway<TAB>2,3 to Eastern Parkway.
\^<TAB>\^<TAB>Telephone<TAB>212-638-5000
_
T{
New York
Historical Society
T}<TAB>T{
All the original paintings for
Audubon's
.I
Birds of America
.R
```

Input Cont'd:

```
are here, as are exhibits of American
decorative arts, New York
history, Hudson River school
paintings, carriages, and glass
paperweights.
T} <TAB> Hours <TAB> T{
Tues-Fri & Sun, 1-5; Sat 10-5
T}
\^ <TAB> \^ <TAB> Location <TAB> T{
Central Park West & 77th St.
T}
\^ <TAB> \^ <TAB> Admission <TAB> Free
\^ <TAB> \^ <TAB> Subway <TAB> AA to 81st St.
&\^ <TAB> \^ <TAB> Telephone <TAB> 212-873-3400
.TE
```

Output:

Some Interesting Places			
Name	Description	Practical Information	
American Museum of Natural History	The collections fill 11.5 acres (Michelin) or 25 acres (MTA) of exhibition halls on four floors. There is a full-sized replica of a blue whale and the world's largest star sapphire (stolen in 1964).	Hours Location Admission Subway Telephone	10-5, ex. Sun 11-5, Wed. to 9 Central Park West & 79th St. Donation: $1.00 asked AA to 81st St. 212-873-4225
Bronx Zoo	About a mile long and .6 mile wide, this is the largest zoo in America. A lion eats 18 pounds of meat a day while a sea lion eats 15 pounds of fish.	Hours Location Admission Subway Telephone	10-4:30 winter, to 5:00 summer 185th St. & Southern Blvd, the Bronx. $1.00, but Tu,We,Th free 2, 5 to East Tremont Ave. 212-933-1759
Brooklyn Museum	Five floors of galleries contain American and ancient art. There are American period rooms and architectural ornaments saved from wreckers, such as a classical figure from Pennsylvania Station.	Hours Location Admission Subway Telephone	Wed-Sat, 10-5, Sun 12-5 Eastern Pkway & Washington Ave. Brooklyn. Free 2,3 to Eastern Parkway. 212-638-5000
New York Historical Society	All the original paintings for Audubon's *Birds of America* are here, as are exhibits of American decorative arts, New York history, Hudson River school paintings, carriages, and glass paperweights.	Hours Location Admission Subway Telephone	Tues-Fri & Sun, 1-5; Sat 10-5 Central Park West & 77th St. Free AA to 81st St. 212-873-3400

nroff/troff: Technical Discussion

Introduction

nroff is a text formatter for typewriter-like printers and terminals. A text formatter manipulates text by interpreting special commands that you place between the lines for which you want formatted output. **nroff** allows you to format a variety of documents, including letters, reports, and books. You will learn the basics of **nroff** by using this tutorial, following the examples that it provides.

The prerequisites to benefit from this tutorial are as follows:

- You should know what a file and directory are and how to create them. See *Using the DG/UX System* (069-701035).
- You should know how to use a text editor (for example, **ed** or **vi**). See *Using the DG/UX Editors* (069-701036).
- Some experience with **mm** is helpful, but not necessary, in making the most of this tutorial. See the tutorial "The Macro Package **mm**" in this manual.

"The Formatter **troff**" in this guide describes a related but more powerful set of requests that you can use to prepare text for phototypesetters. For details about using **nroff**, refer to the "**nroff** Technical Discussion" in *Documenter's Tool Kit Technical Summary for the DG/UX System*.

After reading this tutorial, you will be able to control many attributes of your formatted document, including the margins, indentation, hyphenation, characters per line, and lines per page. You also will know how to use number registers, define strings, and create simple personal macros. This tutorial provides several examples of lines before and after formatting; closely compare the input lines to the output lines to solidify your understanding of **nroff**'s workings.

Requesting Space (.sp) and Indentation (.ti, .in)

Suppose you have a file named **file.in** that contains these lines:

```
I'm glad I'm not a pair of ragged claws, scuttling across the floor
of seafood restaurants.
This is the last line of this paragraph.
.sp 2
.ti +5m
This is the first line of a new paragraph.
This is the second line.
This is the third line.
This is the fourth line, and so on.
```

Besides text, **file.in** contains two requests: **.sp** and **.ti**. **.sp 2** requests two lines of space between text lines, and **.ti +5m** requests that the next text line be indented five ems (one em is about the width of the letter *m*). The dot (.) in column one alerts **nroff** to the presence of a line that should be executed rather than printed, that is, a control line. The two lower-case letters that follow this dot specify the control that you want **nroff** to exert. Arguments to requests, like **2** to **.sp** and **+5m** to **.ti**, refine how they work.

Requests are the simplest control lines that the Documenter's Tool Kit offers. Each request performs a single formatting task. Macros, in contrast, combine requests in special ways, and thus do several formatting chores. Later in this tutorial, you will learn how to create and use macros in addition to those that are already available with **mm**.

To format **file.in** and to put the results into a new file, type this command line:

nroff file.in > file.out

You can look at **file.out** on your terminal after the shell prompt returns, or you can send **file.out** to a printer with

cat file.out | lp

or send it directly with

nroff file.in | lp

A printed version looks like this:

```
I'm glad I'm not a pair of ragged claws, scuttling across the floor
of seafood restaurants.  This is the last line of this paragraph.


        This is the first line of a new paragraph.  This is the
second line.  This is the third line.  This is the fourth line, and
so on.
```

Your printer or terminal may insert more or less text on a given line than this example shows. The important thing to notice is that **file.out** does not look the same as **file.in**. Using **nroff**, you have the power to make **file.out** look precisely the way you want. You can command **nroff** to insert more (or fewer) blank lines or to indent more (or fewer) spaces by changing the arguments to these two requests. Try putting three spaces between the paragraphs, and indenting the new paragraph seven ems.

The **.ti** request (temporary indent) indents only the line that follows it; the second and following lines of text return to the left margin. This request is helpful when formatting paragraphs. To move all output lines to the right seven spaces, use **.in** instead of **.ti**; then the indent is more than temporary. The contents of **file.in** looks like this:

```
Here, text is flush left.
.in +7m
Notice that with the indent request,  all
text lines are indented seven ems from the current left
margin.
Notice how this differs from the temporary indent request.
.in 0
Now, text is flush left again.
```

A printed copy of **file.out** appears as follows:

```
Here, text is flush left.
                Notice that with the indent request, all text lines
                are indented seven ems from the current left margin.
                Notice how this differs from the temporary indent
                request.
Now, text is flush left again.
```

Here, you indent lines in output seven ems until you set them flush left by typing **.in 0**. Typing **.in** without an argument sets indentation where it was before you last used **.in**. Initially, **nroff** sets indentation to 0 (flush left), so in the example above, **.in** would work the same as **.in 0**.

Instead of resetting the indentation to 0, you may want to set it to another value. Consider the contents of **file.in**:

```
Here, text is flush left.
.in +7m
With the indent request, all
text lines are indented seven ems from the current left
margin.
.in -3m
Now, text is four ems from the left margin (7 - 3 = 4).
```

A printed copy of **file.out** follows:

```
Here, text is flush left.
                With the indent request, all text lines are indented
                seven ems from the current left margin.
        Now, text is four ems from the left margin (7 - 3 = 4).
```

Line Filling (.nf, .fi) and Line Breaks (.br)

Notice that the number of lines you type in is not the number **nroff** puts out. Besides interpreting control lines and making them disappear from the output, **nroff** rearranges your text, filling the page with tightly formatted output.

nroff fills lines automatically. When line filling is on, words accumulate in a line buffer until it is full, and then the buffer is flushed. **file.in** is as follows:

```
This means that words fill a line,
regardless
of their position on the page as they are input.
.nf
The "no-fill" request
turns off line filling,
making output the same as input.
.fi
Line filling stays off
unless you turn it back on with the fill request.
```

A printed copy of **file.out** appears as follows:

```
This means that words fill a line, regardless of their position on
the page as they are input.
The "no-fill" request,
turns off line filling,
making output the same as input.
Line filling stays off unless you turn it back on with the fill
request.
```

nroff also flushes the line buffer when it finds a line break. As you may have noticed, **.sp** forces a line break and produces lines of space. The **.br** request also forces a new line. Consider this version of **file.in**:

```
An explicit break request
.br
starts a new line
but does not insert a line of space
between the lines of text it separates.
```

file.out prints as follows:

```
An explicit break request
starts a new line but does not insert a line of space between the
lines of text it separates.
```

Hyphenation (.hy, .nh, .hc, .hw)

Notice that when **nroff** fills lines, it only puts out whole, never hyphenated, words. By default, **nroff** does not hyphenate. To turn on hyphenation anywhere in your text, use **.hy**. When you switch on hyphenation, you may put a hyphenation indicator in a text word to specify places where the word should be hyphenated if need be. Set this hyphenation indicator with **.hc**. **file.in** looks like this:

```
.hy
.hc @
How would you use the extremely long
word pneu@mono@ultra@microscopic@silico@volcano@coniosis
in a sentence?
```

The formatted, **file.out** follows:

```
How would you use the extremely long word pneumonoultramicroscopic-
silicovolcanoconiosis in a sentence?
```

From this point on, **nroff** interprets the character "@" as an acceptable place to put a hyphen, if needed. After inserting the hyphen, **nroff** flushes the line buffer and starts a new line. **nroff** hyphenates words not containing the hyphenation indicator wherever it wants. Do not use **.hc** without turning on hyphenation with **.hy**.

If you want to specify particular words to be hyphenated in a particular way, use the request **.hw**:

```
.hy
.hw anti-climax
The resolution of the silly plot is an anticlimax.
This director should be fired.
```

Now, every time **nroff** finds "anticlimax" at the end of a buffer, it tries to hyphenate it the way that you specified, not "an-ticlimax" or "anticli-max." If the word cannot fit on the line the way you have specified it, **nroff** does not try to hyphenate it.

Centering (.ce)

.ce centers as many text lines as its argument specifies. With no argument, **.ce** centers one line. Here's **file.in**:

```
.nf
The centering command is effectively a "no-fill" command,
except that output lines
are centered instead of flush left.
If you use the no-fill command with
.ce 4
the centering command, centering takes charge.
The next three lines and the line preceding are centered.
.fi
But now you have turned on line filling.
What happens to the centering?
The answer is that centering has priority over filling.
```

The printed version, **file.out**, looks like this:

```
The centering command is effectively a "no-fill" command,
except that the output lines
are centered instead of flush left.
If you use the no-fill command with
          the centering command, centering takes charge.
       The next 3 lines and the line preceding are centered.
             But now you have turned on line filling.
                  What happens to the centering?
The answer is that centering has priority over filling.
```

Justification, Unpaddable Space (.ad, .na)

nroff ordinarily gives you even (justified) left and right margins. (**mm** gives you a ragged right margin by default.) To change margin justification, use **.ad**, as this version of **file.in** demonstrates:

```
.ad l
Here, you've given the adjustment request the argument "l".
This tells nroff to justify only the left margin.
Many people prefer a ragged right margin.
.ad b
With the "b" argument, .ad justifies both left and right margins.
When there is an even right and left margin,
nroff pads the line by expanding spaces.
This may produce text alignment unpleasing to the eye.
.na
The "no adjustment" request turns right justification off (that is, the
left margin is justified, but the right margin is not).
```

The printed versions appear as follows:

> Here, you've given the adjustment request the argument "l". This tells nroff to justify only the left margin. Many people prefer a ragged right margin. With the "b" argument, .ad justifies both left and right margins. When there is an even right and left margin, nroff pads the line by expanding spaces. This may produce text alignment unpleasing to the eye. The "no adjustment" request turns right justification off (that is, the left margin is justified, but the right margin is not).

One way to adjust the right margin and maintain a line pleasing to the eye is to specify a space that **nroff** cannot expand during justification. To do this, type a backslash followed by a space, "\ " (an unpaddable space), at places **nroff** had padded. The backslash is an escape character; you will find out more about this below.

An alternative to using unpaddable spaces is to request that some seldom-used character, such as a tilde (~), be translated into a space on output. To do this, use the translate request

.tr ~

(that is, dot tr space tilde space). If you find that you need a tilde later in the output, turn it back into a tilde it by inserting this line:

.tr ~~

(dot tr space tilde tilde). Later, you may restore the tilde as an unpaddable space by repeating **.tr ~**, but only after a line break or after **nroff** outputs the line containing the tilde.

What do you suppose happens to the text below when you format it?

```
.ad b
.ce 2
What request wins out?
Will there be adjustment, or centering?
```

Remember what happened when **.fi** competed with **.ce**? Here as there, lines after **.ce** are centered, despite the request for adjustment. After that, left and right margins will be adjusted until you change adjustment or until you use **.na**.

Setting Tabs (.ta)

nroff automatically sets tab stops every eight ens from the current indent, but you can change these stops with **.ta**. Here's **file.in**:

```
.ta 0.5i 1.5i 2.5i 3.0i
The next line contains tabs; the tab request
places the tab stops at particular places:
        Here    is      a       line with tabs.
.ta 1.5i 2.5i 3.0i 3.5i
The next line also contains tabs, but the tab request places the
stops differently from above:
        Here    is      a       line with tabs.
```

The file comes off the printer looking like this:

```
The next line contains tabs; the tab request places the tab stops
at particular places:        Here          is            a
    line with tabs.  The next line also contains tabs, but the tab
request places the stops differently from above:
                        Here          is     a      line with tabs.
```

These tab stops are left-justified, but you can set up right-justified tab stops or centered tab stops, too. For details about how to do this, refer to the chapter "**nroff/troff** Technical Discussion" in the *Documenter's Took Kit Technical Summary for the DG/UX System*.

If you want to position numbers, or if you need more complicated columnar layout, use **tbl**, which is described in the chapter "The Preprocessor **tbl**" in this guide.

Selecting a Font (.ft)

nroff is frequently used with mechanical printers like daisy-wheel printers, which produce documents by striking pre-cast characters as they turn on a wheel. Using such a printer, **nroff** can provide three distinctions among fonts. It can provide a regular font by default (**.ft R** or **\fR**); it can represent an italic font by having the printer underline (**.ft I** or **\fI**); finally, it can provide a bold version of the regular font by having the printer back up and overstrike characters (**.ft B** or **\fB**). **nroff** thus understands three fonts—regular, italics, and bold—even when it is used with a basic mechanical printer. When **nroff** is used with a more advanced printer, such as a laser printer or sophisticated dot matrix printer, it can provide a more pleasing version of regular (or roman), italics, and bold:

abcdefghijklmnopqrstuvwxyz 0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ

To switch fonts, use the **.ft** request: **.ft B** for bold, **.ft I** for italic, and **.ft R** for roman. To return to the previous font, whatever it was, use either **.ft P** or **.ft** with no argument. Once you change fonts with **.ft** though, **nroff** uses the font that you specify until you change fonts again.

Another way to italicize text is to use the **.ul** request. Depending on your printer, **.ul** underlines the next input line or italicizes it. Follow **.ul** with an argument that requests the number of input lines to be italicized, for example **.ul 3**; otherwise, only the line that follows the request is italicized (much the same way that **.ti** indents only the line that follows it).

Fonts also can be changed within a line or word with the escape sequence **\f**. Consider the contents of this file:

```
\fBbold\fIface\fR text
```

A printed version looks like this:

bold*face* text

An escape sequence is a special in-line command that begins with the escape character \ (backslash). This character tells **nroff** that what comes next is special; thus **f** is interpreted as "font" instead of as the letter "f."

To avoid losing the last font requested after each in-line change, restore it with the escape sequence **\fP**, since **nroff** remembers only the last font called. For example, in the next line, the last **\fP** restores the font to whatever the font was before **\fB**:

```
\fBbold\fP\fIface\fP\fR text\fP
```

In this next example, the **\fP** restores the font to italic:

```
\fBbold\fIface\f1 text\fP
```

Margins (.po), Line Length (.ll)

If you are not content with the page dimensions that **nroff** gives you, you can change them. **file.in** looks like this:

```
You can change the left margin with .po, which stands for "page offset."
Here, nroff adds one inch to the existing left
margin to determine the new margin.
.po +1i
Here's another line of text.
Once you change the margin, any indentation is relative to the new value.
To restore the previous left margin, type .po without an argument
.po
```

A formatted version looks like this:

```
You can change the left margin with .po, which stands for "page offset."
Here, nroff adds one inch to the existing left margin to determine the new
margin.
                Here's another line of text.  Once you change the margin,
                any indentation is relative to the new value.  To restore
                the previous margin, type .po without an argument.
```

Even though **.po** may appear to do the same thing that **.in** does, it doesn't. The formatting request **.in** indents from the current left margin, while **.po** changes the current left margin.

Look carefully at this last example. Notice that part of **file.in** before the second **.po** is not offset three spaces on output. Remember, **nroff** works with line buffers, not lines as you have typed them. After the second **.po**, the next buffer, not necessarily the next text line, that **nroff** flushes is the first to obey this request.

Also notice that **nroff** translates the escape sequence **\&** into a character that does not print. **\&** is useful when you want to treat a control line as text rather than as something to be executed. Putting this non-printable sequence in columns one and two, before the dot in a control line, nullifies the line's control (as the last example shows). Use **\&** consistently before any control sequence that you want to nullify,

regardless of its current position on a line since when you edit text, the position of words or characters may change in unexpected ways.

The **.ll** request changes "line length." Here's **file.in**:

```
.ll +3
You can change the left margin with .po, which stands for "page offset."
Here, nroff adds one inch from the existing left
margin to determine the new margin.
.po +1i
Here's another line of text.
If you change both the line length \f2and\fP the left margin, you get
different results than if you simply change the margin.
.po
```

The formatted version looks like this:

Change the left margin with .po, which stands for "page offset." Here, nroff adds one inch from the existing left margin to determine the new margin.

Here's another line of text. If you change both the line length *and* the left margin, you get different results than if you simply change the margin.

As you can see, the line length has in fact extended the right margin. Thus, to decrease the right margin, you increase the line length with **.ll**.

Notice that some of the requests covered so far take alphabetic arguments and some take numeric arguments, preceded or not by a plus or a minus sign. To know what's appropriate for any given request, check the "**nroff/troff** Technical Discussion" in the *Documenter's Took Kit Technical Summary for the DG/UX System*.

For now, consider the use of **+** and **−** before a number. These symbols change the previous setting by the amount you specify, rather than by just overriding it. The distinction is important: **.ll +3** makes lines three characters longer; **.ll 3** makes them three characters long.

Page Length (.pl), Page Breaks (.bp), and Page Numbers (.pn)

Now you have control of the width of your printed page, but what about the length? By default, **nroff** gives you a page 11 inches long. If you want to change the page length, that is, change the amount of space that **nroff** leaves between the text and the physical top and bottom of the page, use **.pl**.

.pl +1i

Here, **.pl** has an argument that consists of a number and a letter. This letter corresponds to a scale. If you do not specify **i** (which stands for inches) and simply type **.pl +1**, **nroff** assumes that you want to increase the present page length by one line space.

You also may specify units for **.ll**, **.po**, **.in**, and **.ti**. The default unit for **.in** and **.ti**, as for most horizontally oriented commands, is ems. (Remember, one em is roughly the size of the character *m*.)

If you want to start a new page, use **.bp**, which stands for begin page. The input file looks like this:

```
And so, in conclusion, and so on.
This is the last sentence of a paper.
.bp
.ce
REFERENCES
```

.pn stands for "page number." The next page, when it occurs, will have the page number that you specify as the argument to this request. Pages that follow will also increment from this new page number.

Keeping Lines Together (.ne)

At times you will want to prevent certain lines from being split across pages. Use **.ne** to tell **nroff** the number of lines of text that you want kept together. Here's an input file using **.ne**:

```
.nf
.ne 4
My address is:
John Smith
1956 Malcolm Road
Hometown, USA
.fi
```

would print the four lines of text on the next page if there was not enough room for all of them on the current page.

Defining a String (.ds)

A string is a named collection of characters not including a newline character. Once you have defined a string with **.ds**, you can use the string name as shorthand for its contents. The following is **file.in**:

```
.ds sG string
Defining your own \*(sG is convenient when
you use a particular word
or sequence of characters many times.
To define a \*(sG, type .ds, then the string name, and then its
definition.
Note that \*(sG is replaced by its definition, the word "string,"
throughout this paragraph when you format it.
How you interpolate a \*(sG depends on
whether the \*(sG name is one or two characters long.
If the \*(sG is one letter long, type "\*"
and then the \(*sG name.
If the \*(sG is two letters long, type "\*("
and then the \*(sG name.
```

The processed file looks like this:

```
Defining your own string is convenient when you use a particular
word or sequence of characters many times.  To define a string,
type .ds, then the string name, and then its definition.  Note that
string is replaced by its definition, the word "string," throughout
this paragraph when you format it.  How you interpolate a string
depends on whether the string name is one or two characters long.
If the string is one letter long, type "\*" and then the string
name.  If the string is two letters long, type "\*(" and then the
string name.
```

Remember that a backslash tells **nroff** that what follows is special in some way. Escape sequences allow in-line control of formatting, such as the interpolation of strings. The backslash begins all escape sequences like *****. The "**nroff/troff** Technical Discussion" in the *Documenter's Took Kit Technical Summary for the DG/UX System* lists and describes all available escape sequences. Typing **\e** tells **nroff** to

interpret \ as the character, backslash, not as the beginning of an escape sequence.

If you must begin a string with blanks, define it as follows:

```
.ds xx "     text
```

The double quote signals the beginning of a definition. There is no need for a trailing quote; the end of the line ends the string.

A string may be several lines long; if **nroff** encounters a \ at the end of any line of the string definition, the backslash is thrown away and the next line added to the current one. So you can create a long string simply by using the backslash like this:

```
.ds xx this \
is a very \
long string
```

Using Number Registers (.nr)

Number registers, like strings, can be useful in setting up a document that you can change easily later. **nroff** can do arithmetic with these number registers, which hold numeric values that control aspects of output style.

Like strings, number registers have one or two character names. They are set by the **.nr** command and can be used anywhere in your input by typing **\n** and then the name (for a one-character register name) or **\n(** and the name (for a two-character register name).

There are many predefined number registers maintained by **nroff**, among them **%** for the current page number; **dy**, **mo**, and **yr** for the current day, month, or year; and **.f** for the current font (which is a number from 1 to 3: 1 for roman, 2 for italic, and 3 for bold). Any of the predefined registers listed and described in the "**nroff/troff** Technical Discussion" in the *Documenter's Took Kit Technical Summary for the DG/UX System* may be used in computations, but some, like **.f**, cannot be changed by **.nr**. The following example puts the page number and the current date in a page title (**.tl**).

```
.tl 'John Smith'\n%'\n(mo-\n(dy-\n(yr'
```

Titles are easy; the whole argument to **.tl** appears as the next line of output. The first part of the argument (**John Smith**) appears in the left-hand corner of the page, the second part appears centered, and the last part appears right justified, like this:

```
John Smith                    21                    5-3-89
```

Here's another example using **nroff** number registers. **file.in** looks like this:

```
.in (\n(.l+\n(.i)/2
This starts lines in the center of the page, regardless of line length.
The request subtracts the current indent (contained in the number
register \f3.i\f1) from the current line
length (contained in the number register \f3.l\f1),
divides the result by two, and indents by that amount.
.in
If you do something like this, you might want to put
indentation back to the left margin at some point.
```

The formatted version looks like this:

This starts lines in the center of the page, regardless of line length. The request subtracts the current indent (contained in the number register **.i**) from the current line length (contained in the number register **.l**), divides the result by two, and indents by that amount.

If you do something like this, you might want to put indentation back to the left margin at some point.

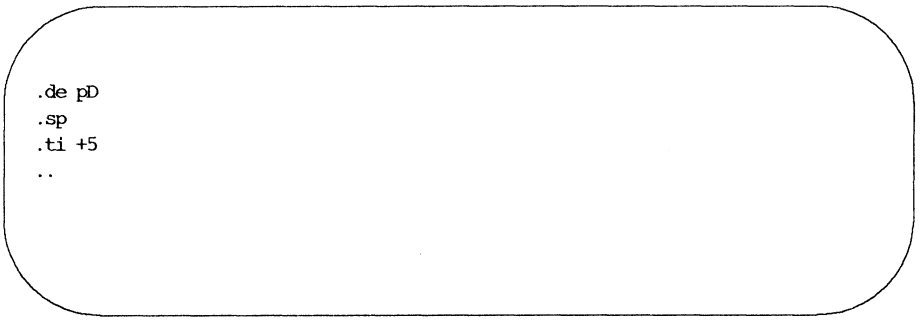
Creating a Simple Macro (.de)

A macro is a shorthand notation similar to a string; it names a collection of requests. When would you want to use a macro rather than a request?

Suppose you want to format every paragraph in a document differently, some with two spaces between them, some indented, some not. Here, it would be reasonable to use requests, since they provide that degree of flexibility. On the other hand, if you wanted to format paragraphs uniformly, you should use a macro. **mm** provides a collection of pre-defined macros that you can use to format several types of documents. However, if you do not want to use **mm** for some reason, you may write your own macros.

For example, in the Sampler file **nroff.letter**, two requests format every paragraph: one request puts a space between the paragraphs (**.sp**) and the other indents the first line five spaces (**.ti +5**). To create your own macro to do the job of these two requests, use the **.de** (for define) request.

You can call your paragraph formatting macro **.pD** (for paragraph definition). Here is how you use **.de** to create **.pD**:



```
.de pD
.sp
.ti +5
..
```

The control line **..** closes the macro definition. You can define macros anywhere in your file that you wish, but it is better to keep all macro definitions at the beginning of your file, for easy maintenance.

After you define your macro, you can call it by name

.pD

and it does the tasks specified by the two requests that it incorporates.

Here is a macro that starts a new page and centers the macro's argument at the top of that particular page:

```
.de nM                          \"new page mast
.bp                             \"begin a new page
.sp 2                           \"two lines of space
.ce                             \"center the next line
\\$1
.sp 3                           \"three lines of space
..
```

Inside this macro definition, the string \\$1 refers to the first argument that you give **.nM** (for example, **.nM REFERENCES**), placing it after **.ce**. Thus, the argument that you give **.nM** gets centered. This centered mast is placed two blank lines down from the top of the page, and then three blank lines are put out.

You may wonder what happens to the words **new page mast**, and so on, inside this macro. **nroff** throws anything after \" away, and then goes to the next line. **nroff** recognizes \" as the beginning of a comment. (Use spaces instead of tabs to set off comments.)

Here's a more elaborate paragraph macro.

```
.de nG                          \"new paragraph
.ft R                           \"roman font
.sp                             \"one line of space
.ne 3i                          \"need three inches
.in 0                           \"flush left
.ti +6                          \"indent next line six spaces
..
```

In this macro, **nroff** loads roman font, puts out a space, sees if there are three inches

of space left on the page (if not, it skips to the next page), sets the indent to the left margin, and then indents the first line six spaces.

Why go to the trouble of setting the indent flush left and then indenting six spaces? Why not simply indent six spaces? You never know where your text has been. Earlier in your file, you may have made a request such as **.in +10**. The line **.in 0** resets indentation and puts the following text at the current left margin. Similarly, loading roman font is a way of ensuring that if you have forgotten to restore roman earlier, you have set things right with this new paragraph.

Notice that these macro names consist of a lower-case letter followed by an upper-case letter. It is good practice to stick to this pattern so that you do not accidentally redefine an **mm** macro.

Moving On

If you want to learn more about **nroff**, scan the material in the "**nroff/troff** Technical Discussion" in the *Documenter's Took Kit Technical Summary for the DG/UX System* and find a request that you think would be useful. Read the material, and then experiment with the request. For example, take what you have learned from this tutorial and explore more complicated uses of number registers and strings.

The **troff** tutorial in this guide explains more complicated requests and macros you can define yourself. As mentioned before, **troff** is a more powerful set of requests that provide precise phototypesetting capabilities.

Introduction

nroff and **troff** are text processors provided with the Documenter's Tool Kit Software. They accept lines of text interspersed with lines of format control information and process the text into a printable, paginated document having a user-designed style. **nroff** and **troff** offer unusual freedom in document styling, including the following: arbitrary style headers and footers; arbitrary style footnotes; multiple automatic sequence numbering for paragraphs, sections, and other user-defined blocks of text; multiple column output; dynamic font and point-size control; arbitrary horizontal and vertical local motions; and a group of automatic overstriking, bracket construction, and line drawing functions.

nroff and **troff** are highly compatible with each other, and it is almost always possible to prepare input acceptable to both. Conditional input is provided that enables the user to embed input expressly destined for either program. **troff** formats text suitable for phototypesetters and printers capable of producing digital-typographical output.

The general form of invoking **nroff** or **troff** at the command level is

nroff *option(s) file(s)* | *printer*
or
troff *option(s) file(s)* | *typesetter*

where *option(s)* represents any of a number of options and option arguments, and *file(s)* represents the file containing the document to be formatted. An argument consisting of a single minus (**–**) is taken to be a file name corresponding to the standard input. If no *file* is given, input is taken from the standard input. The options, which may appear in any order so long as they appear before the file name(s), are

Option	Effect
–e	(**nroff** only.) Produce equally-spaced words in adjusted lines, using full terminal resolution.
–h	(**nroff** only.) Use output tabs during horizontal spacing to speed up output as well as to reduce output byte count. Device tab settings are assumed to be every 8 nominal character widths, as are the settings of input (logical) tabs.
–i	Read standard input after the input files are exhausted.

−m*name* Prepend the macro file **/usr/lib/tmac.***name* to the input *files*. **nroff** and **troff** can accept several **−m** options on the command line causing all macro packages thus named to be read in turn. The possible macro packages (the memorandum macros, the man macros, the viewgraph macros, and the permuted index macros) would be called, with the following options, respectively: **−mm**, **−man**, **−mv**, and **−mptx**.

−n*N* Number first generated page *N*.

−o*list* Print only pages whose page numbers appear in *list*. *list* consists of comma-separated numbers and number ranges. A number range has the form *N*−*M* and means pages *N* through *M*; an initial −*N* means from the beginning to page *N*; and a final *N*− means from *N* to the end.

−q Invoke the simultaneous input-output mode of the **.rd** request.

−r*aN* Set the number register whose (one-character) name is *a* to *N*.

−s*N* Stop every *N* pages. **nroff** will halt prior to every *N* pages (default *N*=1) to allow paper loading or changing, and will resume upon receipt of a new-line. **troff** will stop the phototypesetter every *N* pages, produce a trailer to allow for changing of paper, and will resume after the phototypesetter START button is pressed. In **nroff** the ASCII BEL character will sound when stopping between pages.

–T*tty_type* Specifies the name of **nroff** terminal type, *tty_type*.

Currently defined names are

2631	Hewlett-Packard 2631 printer in regular mode
2631-c	Hewlett-Packard 2631 printer in compressed mode
2631-e	Hewlett-Packard 2631 printer in expanded mode
300	DASI-300 printer
300-12	DASI-300 terminal set to 12-pitch (12 characters per inch)
300s	DASI-300s printer (**300S** is a synonym)
300s-12	DASI-300s printer set to 12-pitch (12 characters per inch) (**300S-12** is a synonym)
37	Teletype Model 37 terminal (default)
382	DTC-382
4000a	Trendata 4000a terminal (**4000A** is a synonym)
450	DASI-450 (Diablo Hyterm) printer
450-12	DASI-450 terminal set to 12-pitch (12 characters per inch)
832	Anderson Jacobson 832 terminal
8510	C.ITOH printer
lp	generic name for printers that can underline and tab. (All text using reverse linefeeds, such as those having tables, that is sent to **lp** must be processed with **col**.)
tn300	GE Terminet 300 terminal
X	printers equipped with TX print train
605x	Data General 605x terminals

(Check with your system administrator for a list of locally supported devices.)

In **troff**, the **–T** option may be used to specify the output device. In addition, it is possible to set the environment variable "TYPESETTER." The output device presently supported for **troff** is the Imagen Imprint-10 (**–Ti10**)

–u*N* (**nroff** only.) Set the emboldening strike-count (number of character overstrikes) for the bold font (normally mounted at the third font position) to be *N*. If option is used without *N*, the number of overstrikes is assumed to be zero. Note that it is not possible to turn off the emboldening in **nroff** if the overstriking is controlled locally by the printing device (e. g., DASI 300s). See the **.bd** request in Section 2.3 and the **.b** number register in Section 4 for a

fuller treatment of emboldening.

−z Suppress formatted output. Only diagnostics and messages from **.tm** requests will occur.

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

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

nroff −o4, 8−10 −T300−12 −m*abc* **file1 file2**

requests formatting of pages 4, 8, 9, and 10 of the document contained in files named **file1** and **file2**, specifies the output terminal as a DASI 300 in 12-pitch, and invokes the macro package *abc*.

Various pre- and postprocessors are available for use with **nroff** and **troff**. These include the preprocessors for writing equations, **neqn** and **eqn** (for **nroff** and **troff** respectively); the preprocessor for making tables, **tbl**; the preprocessor for drawing pictures, **pic**, and the **grap** preprocessor for presenting statistics in a graph. A reverse-line post-processor, **col**, is available for multiple-column **nroff** output on terminals without reverse-line ability; **col** expects the Teletype Model 37 escape sequences that **nroff** produces by default. The finished version of a document typeset with **troff** is most frequently sent to a phototypesetter:

grap *file(s)* | **pic** | **tbl** | **eqn** | **troff** *option(s)* | *typesetter*

The first pipe (|) indicates the redirecting of **grap**'s output to **pic**'s input; the second pipe shows **pic**'s output being redirected to **tbl**'s input; the third pipe shows **tbl**'s output flowing into **eqn**'s input; the fourth indicates the piping of **eqn**'s output to **troff**'s input. Finally, the accumulated output from these processes is piped to a postprocessor that interprets **troff**'s output graphics language for the output device.

tc is a phototypesetter-simulator postprocessor, which enables you to view **troff** output on a Tektronix 4014 terminal. The syntax for its usage is as follows:

grap *file(s)* | **pic** | **tbl** | **eqn** | **troff** *option(s)* | **tc**

The remainder of this manual consists of a "Summary" followed by "A Technical Description of **nroff/troff**." The numbered sections of the "Summary" correspond to numbered sections of the "Technical Description," in which the individual **nroff/troff** subcomponents are covered in detail.

Summary

This summary presents abstracts of requests, escape sequences, number registers, and predefined strings. It gives essential information about these subcomponents such as syntax usage, value of arguments, and default scaling of parameters. The following is a key to some of the notation:

Notes:

B	Request normally causes a break.
D	Mode or relevant parameters associated with current diversion level.
E	Relevant parameters are a part of the current environment.
O	Must stay in effect until logical output.
P	Mode must be still or again in effect at the time of physical output.
T	Parameters are typesetter- or printer-dependent.
v,p,m,u	Default scale indicator; if not specified, scale indicators are ignored.
;	Values separated by ";" are for **nroff** and **troff** respectively.
†	Requests marked with "†" have no effect in **nroff**.
‡	Requests marked with "‡" cause a line break, like that caused by **.br**. Invoking them with the control character "´" (instead of ".") will suppress that break function.

1. General Explanation

(*This topic is covered in section 1 of the "Technical Description" below.*)

2. Font and Character Size Control

Request Form	Initial Value	If No Argument	Notes	Explanation
.ps ±*N*	10 point	previous	E,†	Point size; also **\s**±*N*.
.ss *N*	12/36 em	ignored	E,†	Space-character size set to *N*/36 em.
.cs *FNM*	off	-	P,†	Constant character space (width) mode (font *F*).
.bd *F N*	off	-	P	Embolden font *F* by *N*–1 units.
.bd S *F N*	off	-	P	Embolden Special Font when current font is *F*.
.ft *F*	Roman	previous	E,T	Change to font *F* = *x*, *xx*, or 1-*N*. Also **\f***x*, **\f(***xx*, **\f***N*.

.fp *N F*	-	ignored	T	Font named *F* mounted on physical position $1 \leq N$.

3. Page Control

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

4. Text Filling, Adjusting, and Centering

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

5. Vertical Spacing

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

6. Line Length and Indenting

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

7. Macros, Strings, Diversion, and Position Traps

Request Form	Initial Value	If No Argument	Notes	Explanation
.de *xx yy*	-	*.yy*=..	-	Define or redefine macro *xx;* end at call of *yy*.
.am *xx yy*	-	*.yy*=..	-	Append to a macro.
.ds *xx string*	-	ignored	-	Define a string *xx* containing *string*.
.as *xx string*	-	ignored	-	Append *string* to string *xx*.
.rm *xx*	-	ignored	-	Remove request, macro, or string.
.rn *xx yy*	-	ignored	-	Rename request, macro, or string *xx* to *yy*.
.di *xx*	-	end	D	Divert output to macro *xx*.
.da *xx*	-	end	D	Divert and append to *xx*.

.wh *N xx*	-	-	**v**	Set location trap; negative is with respect to page bottom.
.ch *xx N*	-	-	**v**	Change trap location.
.dt *N xx*	-	off	D,**v**	Set a diversion trap.
.it *N xx*	-	off	E	Set an input-line count trap.
.em *xx*	none	none	-	End macro is *xx*.

8. Number Registers

Request Form	Initial Value	If No Argument	Notes	Explanation
.nr *R* ±*N M*	-	-	**u**	Define and set number register *R*; auto-increment by *M*.
.af *R c*	Arabic	-	-	Assign format to register *R* (*c*=**1, i, I, a, A**).
.rr *R*	-	-	-	Remove register *R*.

9. Tabs, Leaders, and Fields

Request Form	Initial Value	If No Argument	Notes	Explanation
.ta *Nt* ...	0.8;0.5in	none	E,**m**	Tab settings; left type, unless *t*=**R** (right) or *t*=**C** (centered).
.tc *c*	none	none	E	Tab repetition character.
.lc *c*	.	none	E	Leader repetition character.
.fc *a b*	off	off	-	Set field delimiter *a* and pad character *b*.

10. Input and Output Conventions and Character Translations

Request Form	Initial Value	If No Argument	Notes	Explanation
.ec *c*	\	\	-	Set escape character.
.eo	on	-	-	Turn off escape character mechanism.
.lg *N*	-; on	on	-	Ligature mode on if *N*>0.
.ul *N*	off	*N*=1	E	Underline (italicize in **troff**) *N* input lines.
.cu *N*	off	*N*=1	E	Continuous underline in **nroff**; like **.ul** in **troff**.
.uf *F*	Italic	Italic	-	Underline font set to *F* (to be switched to by **.ul**).
.cc *c*	.	.	E	Set control character to *c*.
.c2 *c*	'	'	E	Set no-break control character to *c*.
.tr *abcd....*	none	-	O	Translate *a* to *b*, etc., on output.

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

Vertical Motion	Effect in **troff**	Effect in **nroff**
\v'*N***'**	Move distance *N*	
\u	½ em up	½ line up
\d	½ em down	½ line down
\r	1 em up	1 line up

Horizontal Motion	Effect in **troff**	Effect in **nroff**
\h'*N***'**	Move distance *N*	
****(space)	Unpaddable space-size space	
\0	Digit-size space	
|	1/6 em space	ignored
\^	1/12 em space	ignored

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

Escape Sequence	Function
\o'*string*'	Automatically centered overstriking of characters named in *string*.
\z*c*	Zero-width spacing following character *c* to produce overstruck characters.
\b'*string*'	Vertically pile characters named in *string*.
\l'*Nc*'	Horizontal line drawing function. Optional character *c* may be named to travel to the right a distance *N*. Travel to the left may be specified with a negative *N*.
\L'*Nc*'	Vertical line drawing function. Optional character *c* may be specified to travel downward a distance *N*. Upward distance may be specified with a negative *N*.

13. Hyphenation.

Request Form	Initial Value	If No Argument	Notes	Explanation
.nh	no hyphen	-	E	No hyphenation.
.hy *N*	no hyphen	hyphenate	E	Hyphenate; *N* = mode.
.hc *c*	\%	\%	E	Hyphenation indicator character *c*.
.hw *word1 ...*			ignored	-Exception words.

14. Three Part Titles.

Request Form	Initial Value	If No Argument	Notes	Explanation
.tl *'left'center'right'*		-	-	Three part title.
.pc *c*	%	off	-	Page number character.
.lt ±*N*	6.5 in	previous	E,**m**	Length of title.

15. Output Line Numbering.

Request Form	Initial Value	If No Argument	Notes	Explanation
.nm $\pm N$ *M S I*		off	E	Number mode on or off, set parameters.
.nn *N*	-	*N*=1	E	Do not number next *N* lines.

16. Conditional Acceptance of Input

Request Form	Initial Value	If No Argument	Notes	Explanation
.if *c anything*		-	-	If condition *c* true, accept *anything* as input, for multi-line use \{*anything*\}.
.if !*c anything*		-	-	If condition *c* false, accept *anything*.
.if *N anything*		-	**u**	If expression $N > 0$, accept *anything*.
.if !*N anything*		-	**u**	If expression $N \leq 0$, accept *anything*.
.if *'string1'string2' anything*			-	If *string1* identical to *string2*, accept *anything*.
.if !*'string1'string2' anything*			-	If *string1* not identical to *string2*, accept *anything*.
.ie *c anything*		-	**u**	If portion of if-else; all above forms (like **if**).
.el *anything*		-	-	Else portion of if-else.

17. Environment Switching.

Request Form	Initial Value	If No Argument	Notes	Explanation
.ev *N*	*N*=0	previous	-	Environment switched (push down).

18. Insertions from the Standard Input

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

19. Input/Output File Switching

Request Form	Initial Value	If No Argument	Notes	Explanation
.so *file*	-	-	-	Switch source file (push down).
.nx *file*	-	end-of-file	-	Next file.
.cf *file*	-	-	-	Copy file.
.lf *N file*	-	-	-	change line number and file name.
.pi *program*		-	-	Pipe output to *program*.

20. Miscellaneous

Request Form	Initial Value	If No Argument	Notes	Explanation
.mc *c N*	-	off	E,**m**	Set margin character *c* and separation *N*.
.tm *string*	-	new-line	-	Print *string* on terminal (UNIX standard message output).
.ig *yy*	-	*.yy*=..	-	Ignore till call of *yy*.
.pm *t*	-	all	-	Print macro names and sizes; if *t* present, print only total of sizes.
.fl	-	-	B,‡	Flush output buffer.
.ab *text*	-	user abort	-	Terminates processing immediately.

.sy *cmd args-* - - *cmd* executed but output not captured.

21. Output and Error Messages

(This topic is covered in section 21 of the "Technical Description," which follows.)

22. Request and Section Number Cross Reference

23. Escape Sequences for Characters, Indicators, and Functions

Section Reference	Escape Sequence	Meaning
10.1	****	\ (to prevent or delay the interpretation of \)
10.1	**\e**	Printable version of the current escape character.
2.1	**\'**	´ (acute accent; typographically equivalent to **\(aa**)
2.1	**\`**	` (grave accent; typographically equivalent to **\(ga**)
2.1	**\-**	− (minus sign in the current font)
7.2	**\.**	. (period or dot; see **.de**)
11.1	**\(***space***)**	Unpaddable space-size space character
11.1	**\0**	Digit width space
11.1	**\\|**	1/6 em narrow space character (zero width in **nroff**)
11.1	**\^**	1/12 em half-narrow space character (zero width in **nroff**)
4.1	**\&**	Non-printing, zero width character
10.6	**\!**	Transparent line indicator
10.7	**.\"**	Beginning of comment
7.3	**\$***N*	Interpret argument $1 \leq N \leq 9$
13	**\%**	Default optional hyphenation character
2.1	**\(***xx*	Character named *xx*
7.1	*****x*, ***(***xx*	Interpret string *x* or *xx*
9.1	**\a**	Non-interpreted leader character
12.3	**\b**'*abc...*'	Bracket building function
4.2	**\c**	Interrupt text processing
11.1	**\d**	Forward (down) 1/2 em vertical motion (1/2 line in **nroff**)
12.4	**\D**	Line-drawing functions
2.2	**\f***x*,**\f(***xx*,**\f***N*	Change to font named *x* or *xx*, or position *N*
8	**\g***x*, **\g(***xx*	Return the format of values stored in general number registers *x* and *xx*.
11.1	**\h**'*N*'	Local horizontal motion; move right *N* (negative left)
2.3	**\H**'*N*'	Height control of characters (does not affect width).
11.3	**\k***x*	Mark horizontal input place in register *x*
12.4	**\l**'*Nc*'	Horizontal line drawing function (optionally with *c*)
12.4	**\L**'*Nc*'	Vertical line drawing function (optionally with *c*)
8	**\n***x*,**\n(***xx*	Evaluate number register *x* or *xx*
12.1	**\o**'*abc...*'	Overstrike characters *a, b, c, ...*
4.1	**\p**	Break and spread output line
11.1	**\r**	Reverse (up) 1 em vertical motion (reverse line in **nroff**)

2.3	**\s***N*, **\s**±*N*	Point-size change function
2.3	**\S**'*n*'	Slant control of characters.
9.1	**\t**	Non-interpreted horizontal tab
11.1	**\u**	Reverse (up) 1/2 em vertical motion (1/2 line in **nroff**)
11.1	**\v**'*N* '	Local vertical motion; move down *N* (negative up)
11.2	**\w**'*string*'	Evaluate and use width of *string*
5.2	**\x**'*N* '	Extra line-space function (negative before, positive after)
12.2	**\z***c*	Print *c* with zero width (without spacing)
16	**\{**	Begin conditional input
16	**\}**	End conditional input
10.7	\(new-line)	Concealed (ignored) new-line
-	*C*	*C*, any character not listed above

The escape sequences ****, **\.**, **\"**, **\$**, *****, **\a**, **\n**, **\t**, and \(new-line) are interpreted in copy mode.

24. Predefined General Number Registers

Section Reference	Register Name	Description
3	**%**	Current page number.
2.2	**.b**	Emboldening factor of current font.
-	**c.**	Input line-number in the current input file. (Contains the same value as the read-only **.c** register.)
-	**.R**	Count of number registers that remain available for use.
11.2	**ct**	Character type (set by width function).
7.4	**dl**	Width (maximum) of last completed diversion.
7.4	**dn**	Height (vertical size) of last completed diversion.
-	**dw**	Current day of the week (1-7).
-	**dy**	Current day of the month (1-31).
15	**ln**	Output line number.
-	**mo**	Current month (1-12).
4.1	**nl**	Vertical position of last printed text base-line.
11.2	**sb**	Depth of string below base line (generated by width function).
11.2	**st**	Height of string above base line (generated by width function).
-	**yr**	Last two digits of current year.

25. Predefined Read-only Number Registers

Section Reference	Register Name	Description
7.3	**.$**	Number of arguments available at the current macro level.
-	**$$**	Identification number (process i.d.) for **nroff** or **troff** processes.
-	**.A**	Set to 1 in **troff** if **−a** option used; always 1 in **nroff**.
-	**.F**	*string* that is the name of the current input file.
11.1	**.H**	Available horizontal resolution in basic units.
-	**.L**	Current line-spacing parameter (see request, **.ls**).

-	**.P**	Set to 1 if the current page is being printed, and zero otherwise.
-	**.T**	Set to 1 if **–T** option used; otherwise set to 0.
11.1	**.V**	Available vertical resolution in basic units.
5.2	**.a**	Post-line extra line-space most recently utilized using **\x**'*N*'.
-	**.c**	Number of lines read from current input file.
7.4	**.d**	Current vertical place in current diversion; equal to **nl**, if no diversion.
2.2	**.f**	Current font as numerical position.
4.1	**.h**	Text base-line high-water mark on current page or diversion.
6	**.i**	Current indent.
-	**.j**	Current adjustment mode and type. Can be saved and later given to the **.ad** request to restore a previous mode.
-	**.k**	Horizontal size of the text portion (without indent) of the current partially collected output line, if any, in the current environment.
6	**.l**	Current line length.
4.1	**.n**	Length of text portion on previous output line.
3	**.o**	Current page offset.
3	**.p**	Current page length.
2.3	**.s**	Current point size.
7.5	**.t**	Distance to the next trap.
4.2	**.u**	Equal to 1 in fill mode and 0 in no-fill mode.
5.1	**.v**	Current vertical line spacing.
11.2	**.w**	Width of previous character.
-	**.x**	Reserved version-dependent register.
-	**.y**	Reserved version-dependent register.
7.4	**.z**	Name of current diversion.

26. Predefined Strings

-	.T	Name of output device.

A Technical Description of nroff/troff

This section explains the subcomponents listed in the summary. The subcomponents can be cross-referenced by section number.

1. General Explanation

1.1. Form of Input

Input consists of text lines, which are destined to be printed, interspersed with control lines, which set parameters or otherwise control subsequent processing. Control lines begin with a control character—normally a period (.), or an acute accent (´) followed by a one or two character name that specifies a basic request or the substitution of a user-defined macro in place of the control line. The control character "´" suppresses the break function—the forced output of a partially filled line—caused by certain requests. The control character may be separated from the request or macro name by white space (spaces and/or tabs) for aesthetic reasons. Names must be followed by either space or a new-line. Control lines with unrecognized names are ignored.

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

1.2. Formatter and Device Resolution

nroff internally uses 240 units/inch, corresponding to the least common multiple of the horizontal and vertical resolutions of various current typewriter-like output devices. Units in **troff** are device-dependent. **troff** rounds horizontal/vertical numerical parameter input to its internal horizontal/vertical resolution. **nroff** similarly rounds numerical input to the actual resolution of the output device indicated by the **–T** option (default Teletype Model 37).

1.3. Numerical Parameter Input

Both **nroff** and **troff** accept numerical input with the appended scale indicators shown in the following table, where *S* is the current type size in points, *V* is the current vertical line spacing in basic units, and *C* is a nominal character width in basic units.

Scale Indicator	Meaning	Number of Basic Units	
		troff	**nroff**
i	Inch		240
c	Centimeter		240×50/127
P	Pica = 1/6 inch		240/6
m	Em = *S* points	machine-	*C*
n	En = Em/2	dependent	*C, same as Em*
p	Point = 1/72 inch		240/72
u	Basic unit		1
v	Vertical line space		*V*
none	Default, see below		

In **nroff**, both the em and the en are taken to be equal to the *C*, which is output-device dependent; frequent values are 1/10 and 1/12 inch. Actual character widths in **nroff** need not be all the same, and characters constructed with predefined strings such as –> (→) are often extra wide.

The scaling for horizontally-oriented control characters, vertically-oriented control characters, and the requests **.nr**, **.if**, and **.ie** are as follows:

Orientation	Measure by Default	Request or Function
Horizontal	Em (**m**)	**.ll, .in, .ti, .ta, .lt, .po, .mc, \h, \l.**
Vertical	Vertical line space (**v**)	**.pl, .wh, .ch, .dt, .sp, .sv, .ne, .rt, \v, \x, \L**
Register-oriented or conditional	Basic unit (**u**)	**.nr, .if, .ie.**
Miscellaneous	Point (**p**)	**.ps, .vs, \H, \s.**

All other requests ignore any scale indicators. When a number register containing an already appropriately scaled number is interpreted to provide numerical input, the unit scale indicator **u** may need to be appended to prevent an additional inappropriate

default scaling. The number, *N*, may be specified in decimal-fraction form, but the parameter finally stored is rounded to an integer number of basic units.

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

```
.sp |3.2c
```

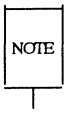
will space in the required direction to 3.2 centimeters from the top of the page.

1.4. Numerical Expressions

Wherever numerical input is expected an expression involving parentheses, the arithmetic operators +, −, /, *, % (mod), and the logical operators **<**, **>**, **<=**, **>=**, = (or ==), & (and), and **:** (or) may be used. Except where controlled by parentheses, evaluation of expressions is left-to-right. In the case of certain requests, an initial + or − is stripped and interpreted as an increment or decrement indicator respectively. In the presence of default scaling, the desired scale indicator must be attached to every number in an expression for which the desired and default scaling differ. For example, if the number register **x** contains 2 and the current point size is 10, then

```
.ll (4.25i+\nxP+3)/2u
```

will set the line length to 1/2 the sum of 4.25 inches + 2 picas + 3 ems.



The use of white space in arithmetic expressions is not permitted. There is no precedence among arithmetic and logical operators. **nroff/troff** expressions do not recognize decimal multipliers or divisors; a high level of precision may be achieved by mixing scales within expressions.

1.5. Notation

Numerical parameters are indicated in this manual in two ways. ±*N* means that the argument may take the form *N*, +*N*, or −*N* and that the corresponding result is to set the affected parameter to *N*, to increment it by *N*, or to decrement it by *N* respectively. Plain *N* means that an initial algebraic sign is not an increment indicator, but merely the sign of *N*. Generally, unreasonable numerical input is either ignored or truncated to a reasonable value. For example, most requests expect to set parameters to non-negative values: exceptions are **.sp**, **.wh**, **.ch**, **.nr**, and **.if**. The requests **.ps**,

.ft, **.po**, **.vs**, **.ls**, **.ll**, **.in**, and **.lt** restore the previous parameter value in the absence of an argument.

Single character arguments are indicated by single lower-case letters and one or two character arguments are indicated by a pair of lower-case letters. Character string arguments are indicated by multi-character mnemonics.

2. Font and Character Size Control

2.1. Character Set

The **troff** character set consists of the so-called Commercial II character set plus a Special Mathematical Font character set. With three exceptions, these ASCII characters are input as themselves, and non-ASCII characters are input in the form **\(***xx* where *xx* is a two-character name. The three ASCII exceptions are mapped as follows:

ASCII Input		Printed by **troff**	
Character	Name	Character	Name
´	acute accent	’	close quote
`	grave accent	‘	open quote
–	minus	-	hyphen

The characters ´, `, and – may be input by \´, \`, and \– respectively or by their names. (See the Table of Special Characters.) ASCII control characters are discussed in the section, "Input Character Translation."

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

2.2. Fonts

Default fonts may differ from device to device. Typically, the fonts will include the following: Times Roman (**R**), Times Italic (**I**), Times Bold (**B**), and the Special Mathematical Font (**S**). The current font may be changed by use of the **.ft** request, or by embedding at any desired point either **\f***x*, **\f(***xx*, or **\f***N* where *x* and *xx* are the name of a mounted font and *N* is a numerical font position. It is not necessary to change to the Special Font; characters on that font are handled automatically.

A request for a named but not mounted font is translated into a request to mount the font at position 0. This position is reserved for such dynamic requests and is otherwise inaccessible. **troff** can be informed that any particular font is mounted by use of the **.fp** request. The list of known fonts is device-dependent. In the subsequent discussion of font-related requests, *F* represents either a one or a two-character font name. The current font is available (as a numerical position) in the read-only number register **.f**. **nroff** understands font control and normally underlines italic characters. (See section 10.3, "Backspacing, Underlining, and Overstriking.")

2.3. Character Size

The available character point sizes depend on the individual printing device. The **.ps** request is used to change or restore the point size. Alternatively, the point size can be changed between any two characters by embedding a **\s***N* at the desired point. (*N* may represent the desired available size or **\s**±*N* may be used to increment or decrement the size by *N*; double-digit increments or decrements are expressed as \s±*NN*.) **\s0** restores the previous size. Requested point size values that are between two legal sizes will yield the closer legal size. The current size is available in the **.s** register.

In **troff** the escape sequence **\H**'*N*' sets the height of a character without affecting its width. *N* can be expressed in absolute values or in relative values of the form ±*N*.

Request Form	Initial Value	If No Argument	Notes	Explanation
.ps ±*N*	10 point	previous	E	Point size set to ±*N*. Alternatively, embed **\s***N* or **\s**±*N*. Any positive size value may be requested; if invalid, the nearest valid size will result, with a maximum size to be determined by the individual printing device. A paired sequence +*N*, –*N* will work because the previous requested value is also remembered. Ignored in **nroff**.
.ss *N*	12/36 em	ignored	E	Space-character size is set to *N*/36 ems. This size is the minimum word spacing in adjusted text. Ignored in **nroff**.
.cs *FNM*	off	-	P	Constant character space (width) mode is set on for font *F* (if mounted); the width of every character will be taken to be *N*/36 ems. If *M* is absent, the

em is that of the character's point size; if *M* is given, the em is *M*-points. All affected characters are centered in this space, including those with an actual width larger than this space. Special Font characters occurring while the current font is *F* are also so treated. If *N* is absent, the mode is turned off. The mode must be still or again in effect when the characters are physically printed. Ignored in **nroff**.

.bd *F N* off - P

The characters in font *F* will be artificially emboldened by printing each one three times, separated by *N*–1 basic units. If *N* is missing the embolden mode is turned off. In **nroff** the default setting is **.bd 3 3**, specifying that characters on the font mounted at position 3 (normally bold) are to be overstruck 3 times (i. e., printed in place a total of 4 times).

The column heads above were printed with **.bd 2 3**. That is, the font stored at font position 2 was emboldened by an offset of 3. The font name itself may be substituted for the font position (e. g. **.bd I 3**). The **–u** command line option may be used to change the number of overstrikes using an argument that is functionally identical to **.bd**'s second argument. (See section 2.3, "Character Size.") The mode must be still or again in effect when the characters are physically printed. This request may affect the contents of the general number register **.b**.

Note that it is not possible to turn off the emboldening in **nroff** if it is being controlled locally by the printing device (e. g., DASI 300s).

.bd S *F N*	off	-	P	The characters in the Special Font will be emboldened whenever the current font is *F*. The mode must be still or again in effect when the characters are physically printed.
.ft *F*	Roman	previous	E	Font changed to *F*. Alternatively, embed **\f***F*. The font name **P** is reserved to mean the previous font.
.fp *N F file*	-	ignored	-	Font position. This is a statement that a font named *F* is mounted on position *N*. It is a fatal error if *F* is not known. Fonts and the possible range of their numerical positions is device-dependent. **.fp** accepts a third optional argument, *file*, which is an alternate version of the font *F*.

.bd can be used to embolden characters, effectively increasing the number of available fonts. This capability of modifying existing fonts to make new ones is enhanced with the **troff** escape sequence, **\S**, used to slant output characters by a number of specified degrees. This escape sequence is stated as **\S'***N***'** where *N* may be any integer, negative or positive. **0** turns slanting off.

3. Page control

Top and bottom margins are not automatically provided; it is conventional to define two macros and to set traps for them at vertical positions 0 (top) and –N (N from the bottom). A pseudo-page transition onto the first page occurs either when the first break occurs or when the first non-diverted text processing occurs. Arrangements for a trap to occur at the top of the first page must be completed before this transition. In the following, references to the current diversion mean that the mechanism being described works during both ordinary and diverted output (the former being considered as the top diversion level). (See section 7.4, "Diversions.")

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

Request Form	Initial Value	If No Argument	Notes	Explanation
.pl ±*N*	11 in	11 in	v	Page length set to ±*N*. The internal limitation is about 75 inches in **troff** and about 136 inches in **nroff**. The current page length is available in the **.p** register.
.bp ±*N*	*N*=1	-	B,‡,v	Break page. The current page is ejected and a new page is begun. If ±*N* is given, the new page number will be ±*N*. Also see request **.ns**.
.pn ±*N*	*N*=1	ignored	-	Page number. The next page (when it occurs) will have the page number ±*N*. A **.pn** must occur before the initial pseudo-page transition to affect the page number of the first page. The current page number is in the **%** register.
.po ±*N*	0; 1 in†	previous	v	Page offset. The current left margin is set to ±*N*. The **troff** initial value provides 1 inch of paper margin. (See section 6, "Line Length and Indenting.") The current page offset is available in the **.o** register.
.ne *N*	-	*N*=1 *V*	D,v	Need *N* vertical space. If the page space needed (*N*) is greater than the distance to the next trap (*D*), then a forward vertical space of size *D* occurs, which will spring the trap. If there are no remaining traps on the page, *D* is the distance to the bottom of the page. If the distance to the next trap (*D*) is less than one vertical line space (**v**), then another line could still be output before the trap is sprung. In a diversion, *D* is the distance to the diversion trap, if any, or is very large.
.mk *R*	none	internal	D	Mark the current vertical place in an internal register (both associated with the

				current diversion level), or in register *R*, if given. See **.rt** request.
.rt ±*N*	none	internal	D,v	Return upward only to a marked vertical place in the current diversion. If ±*N* (relative to the current place) is given, the place is ±*N* from the top of the page or diversion or, if *N* is absent, the place is that marked by a previous **.mk**. Note that the **.sp** request may be used in all cases instead of **.rt** by spacing to the absolute place stored in a explicit register; e.g., using the sequence **.mk** *R* ... **.sp |\n***R***u**. (See section 5.3, "Blocks of Vertical Space.")

4. Text Filling, Adjusting, and Centering

4.1. Filling and Adjusting

During fill mode, words normally are collected from input text lines and assembled into an output text line until some word doesn't fit. An attempt is then made to hyphenate the word in an effort to assemble a part of it into the output line. In adjust mode the spaces between the words on the output line are then increased to spread out the line to the current line length minus any current indent. A word is any string of characters delimited by the space character, the beginning or end of the input line, or by a combination of these. Any adjacent pair of words that must be kept together (neither split across output lines nor spread apart in the adjustment process) can be tied together by separating them with the unpaddable space character "**\ **" (backslash-space). The adjusted word spacings are uniform in **troff** and the minimum interword spacing can be controlled with the **.ss** request. (See section 2, "Font and Character Size Control.") In **nroff**, they are normally nonuniform because of quantization to character-size spaces; however, the command-line option **−e** causes uniform spacing with full output device resolution. Filling, adjustment, and hyphenation can all be prevented or controlled. (See section 13, "Hyphenation.") The text length on the last line output is available in the **.n** register, and text base-line position on the page for this line is in the **nl** register. The text base-line high-water mark (lowest place) on the current page is in the **.h** register.

An input text line ending with ., ?, !, .), ?), or !) is taken to be the end of a sentence, and an additional space character is automatically provided during filling. Multiple inter-word space characters found in the input are retained except for trailing spaces; initial spaces also cause a break.

When filling is in effect, a **\p** may be embedded or attached to a word to cause a break at the end of the word and have the resulting output line spread out to fill the current line length.

A text input line that happens to begin with a control character can be made to not look like a control line by prefacing it with the non-printing, zero-width filler character **\&**. Still another way is to specify output translation of a specific character into the control character using **.tr**. (See section 10.4, "Output Translation.") To suppress the break function of the control character ".", replace it with the single quote, "'".

4.2. Interrupted Text

The copying of an input line in no-fill (i.e., no line filling) mode can be interrupted by terminating the partial line with a **\c**. The next encountered input text line will be considered to be a continuation of the same line of input text. Similarly, a word within filled text may be interrupted by terminating the word (and line) with **\c**; the next encountered text will be taken as a continuation of the interrupted word. If the intervening control lines cause a break, any partial line will be forced out along with any partial word.

Request Form	Initial Value	If No Argument	Notes	Explanation
.br	-	-	B	Break. The filling of the line currently being collected is stopped and the line is output without adjustment. Text lines beginning with space characters and empty text lines (blank lines) also cause a break.
.fi	fill on	-	B,E	Fill subsequent output lines. The register **.u** is 1 in fill mode and 0 in no-fill mode.
.nf	fill on	-	B,E	No-fill. Subsequent output lines are neither filled nor adjusted. Input text lines are copied directly to output lines without regard for the current line length.

.ad *c* | adj,both | adjust | E | Line adjustment is begun. If fill mode is not on, adjustment will be deferred until fill mode is back on. If the type indicator *c* is present, the adjustment type is changed as shown in the following table.

Indicator	Adjust Type
l	adjust left margin only
r	adjust right margin only
c	center
b or **n**	adjust both margins
absent	unchanged

The adjustment type indicator *c* may also be a number obtained from the **.j** register. (See section 25 in the "Summary," "Predefined Read-Only Registers.")

.na | adjust | | E | No-adjust. Adjustment is turned off; the right margin will be ragged. The adjustment type for **.ad** is not changed. Output line filling still occurs if fill mode is on.

.ce *N* | off | *N*=1 | B,E | Center the next *N* input text lines within the current (line-length minus indent). If *N*=0, any residual count is cleared. A break occurs after each of the *N* input lines. If the input line is too long, it will be left adjusted.

5. Vertical Spacing

5.1. Base-line Spacing

The vertical spacing *(V)* between the base-lines of successive output lines can be set using the **.vs** request with a resolution that is device-dependent in both **nroff** and **troff**. *V* must be large enough to accommodate the character sizes on the affected output lines. For the common type sizes (9-12 points), usual typesetting practice is to set *V* to 2 points greater than the point size; **troff** default is 10-point type on a 12-point

spacing (as in this document). The current *V* is available in the **.v** register. Multiple-*V* line separation (e.g., double spacing) may be requested with **.ls**.

5.2. Extra Line-space

If a word contains a vertically tall construct requiring the output line containing it to have extra vertical space before and/or after it, the extra-line-space function **\x**'*N*' can be embedded in or attached to that word. In this and other functions having a pair of delimiters around their parameter (here '), the delimiter choice is arbitrary except that it can't look like the continuation of a number expression for *N*. If *N* is negative, the output line containing the word will be preceded by *N* extra vertical space; if *N* is positive, the output line containing the word will be followed by *N* extra vertical space. If successive requests for extra space apply to the same line, the maximum values are used. The most recently utilized post-line extra line-space is available in the **.a** register.

5.3. Blocks of Vertical Space

A block of vertical space is ordinarily requested using **.sp**, which honors the no-space mode and which does not space past a trap. A contiguous block of vertical space may be reserved using **.sv**.

Request Form	Initial Value	If No Argument	Notes	Explanation
.vs *N*	1/6in;12pts	previous	E,p	Set vertical base-line spacing size *V*. Transient extra vertical space available with **\x**'*N*' (see above).
.ls *N*	*N*=1	previous	E	Line spacing set to ±*N*. *N*–1 *V*s (blank lines) are appended to each output text line. Appended blank lines are omitted, if the text or previous appended blank line reached a trap position.
.sp *N*	-	*N*=1*V*	B,v	Space vertically in either direction. If *N* is negative, the motion is backward (upward) and is limited to the distance to the top of the page. Forward (downward) motion is truncated to the distance to the nearest trap. If the no-space mode is on, no spacing occurs (see **.ns** and **.rs** below).

Description				
.sv *N*	-	*N*=1*V*	v	Save a contiguous vertical block of size *N*. If the distance to the next trap is greater than *N*, *N* vertical space is output. No-space mode has no effect. If this distance is less than *N*, no vertical space is immediately output, but *N* is remembered for later output (see **.os**). Subsequent **.sv** requests will overwrite any still remembered *N*.
.os	-	-	-	Output saved vertical space. No-space mode has *no* effect. Used to finally output a block of vertical space requested by an earlier **.sv** request.
.ns	space	-	D	No-space mode turned on. When on, the no-space mode inhibits **.sp** requests and **.bp** requests without a next page number. The no-space mode is turned off when a line of output occurs, or with **.rs**.
.rs	space	-	D	Restore spacing. The no-space mode is turned off.
Blank text line		-	B	Causes a break and outputs a blank line exactly like **.sp 1**.

6. Line Length and Indenting

The maximum line length for fill mode may be set with **.ll**. The indent may be set with **.in**; an indent applicable to only the next output line may be set with **.ti**. The line length includes indent space but not page offset space. The line-length minus the indent is the basis for centering with **.ce**. The effect of **.ll**, **.in**, or **.ti** is delayed, if a partially collected line exists, until after that line is output. In fill mode the length of text on an output line is less than or equal to the line length minus the indent. The current line length and indent are available in registers **.l** and **.i**, respectively. The length of three-part titles produced by **.tl** is independently set by **.lt**. (See section 14, "Three Part Titles.")

Request Form	Initial Value	If No Argument	Notes	Explanation
.ll ±*N*	6.5 in	previous	E,**m**	Line length is set to ±*N*. In **troff** the maximum (line-length)+(page-offset) is device-dependent.
.in ±*N*	*N*=0	previous	B,E,**m**	Indent is set to ±*N*. The indent is prepended to each output line.
.ti ±*N*	-	ignored	B,E,**m**	Temporary indent. The next output text line will be indented a distance ±*N* with respect to the current indent. The resulting total indent may be zero (equal to current page offset) but may not be less than the current page offset. The temporary indent applies only for the one output line following the request; the value of the current indent (that value stored in the **.i** register) is not changed.

7. Macros, Strings, Diversions, and Position Traps

7.1. Macros and Strings

A macro is a named set of arbitrary lines that may be invoked by name or with a trap. A string is a named string of characters, not including a new-line character, that may be summoned by name at any point. Request, macro, and string names share the same name list. Macro and string names may be one or two characters long and may usurp previously defined request, macro, or string names. Any of these entities may be renamed with **.rn** or removed with **.rm**.

Macros are created by **.de** and **.di**, and appended to by **.am** and **.da**; **.di** and **.da** cause normal output to be stored in a macro. Strings are created by **.ds** and appended to by **.as**. A macro is invoked in the same way as a request; a control line beginning *.xx* will execute the contents of macro *xx* (macro **.xx** may be followed by a maximum of nine arguments): should the macro-define contain requests or escape sequences, these will be executed along with the rest of the defined contents of *xx*. If the macro is defined to contain text, that text will print. The strings defined as *x* and *xx* are inserted at any desired point with *******x*** and *******(***xx*** respectively. String references

and macro invocations may be nested.

7.2. Copy Mode Input Interpretation

During the definition and extension of strings and macros (not by diversion) the input is read in copy mode. The input is copied without interpretation except that

- The contents of number registers, indicated by **\n**, are substituted.
- Strings, indicated by *****x* and *****(xx*, are read into the text.
- Arguments indicated by **\$** are replaced by the appropriate values at the current macro level.
- Concealed new-lines indicated by \(new-line) are eliminated.
- Comments indicated by **\"** are eliminated.
- **\t** and **\a** are interpreted as ASCII horizontal tab and SOH respectively. (See section 9, "Tabs, Leaders, and Fields.")
- **** is interpreted as ****.
- **\.** is interpreted as **"."**.

These interpretations can be suppressed by prepending a ****. For example, since **** maps into a ****, **\\n** will copy as **\n** which will be interpreted as a number register indicator when the macro or string is reread.

7.3. Arguments

When a macro is invoked by name, the remainder of the line is taken to contain up to nine arguments. The argument separator is the space character, and arguments may be surrounded by double-quotes to permit embedded space characters. Pairs of double-quotes may be embedded in double-quoted arguments to represent a single double-quote. If the desired arguments won't fit on a line, a concealed new-line may be used to continue on the next line.

When a macro is invoked the input level is pushed down and any arguments available at the previous level become unavailable until the macro is completely read and the previous level is restored. A macro's own arguments can come into play at any point within the macro with **\$***N*, which introduces the *N*th argument ($1 \leq N \leq 9$) into the macro's processing activity. If an invoked argument doesn't exist, a null string results. For example, the macro *xx* may be defined by

```
.de xx      \"begin definition
```

```
Today is \\$1 the \\$2.
..          \"end definition
```

and called by

```
.xx Monday 14th
```

to produce the text

Today is Monday the 14th.

Notice that the **\$** was concealed in the definition with a prepended ****. This leading backslash protected the argument ($) so that it would be replaced by the argument of the macro when invoked, and not by the argument in effect when this one was being defined. The number of currently available arguments is in the **.$** register.

No arguments are available at the top (non-macro) level in this implementation. Because string referencing is implemented as an input-level push down, no arguments are available from within a string. No arguments are available within a trap-invoked macro.

Arguments are copied in copy mode onto a stack where they are available for reference. The mechanism does not allow an argument to contain a direct reference to a long string (evaluated at copy time), and it is advisable to conceal string references (with an extra ****) to delay interpretation until argument reference time.

7.4. Diversions

Processed output may be diverted into a macro for purposes such as footnote or sidenote processing or determining the horizontal and vertical size of some text for conditional changing of pages or columns. A single diversion trap may be set at a specified vertical position. The number registers **dn** and **dl** respectively contain the vertical and horizontal size of the most recently ended diversion. Processed text that is diverted into a macro retains the vertical size of each of its lines when reread in no-fill mode regardless of the current *V*. Constant-spaced (**.cs**) or emboldened (**.bd**) text that is diverted can be reread correctly only if these modes are again or still in effect at reread time. One way to do this is to embed in the diversion the appropriate **.cs** or **.bd** requests with the transparent mechanism described in section 10.6, "Transparent Throughput."

Diversions may be nested and certain parameters and registers are associated with the current diversion level (the top non-diversion level may be thought of as the 0th diversion level). These are the diversion trap and associated macro, no-space mode, the internally-saved marked place (see **.mk** and **.rt**), the current vertical place (**.d** register), the current high-water text base-line (**.h** register), and the current diversion name (**.z** register).

7.5. Traps

Three types of trap mechanisms are available—page traps, a diversion trap, and an input-line-count trap. Macro-invocation traps may be planted using **.wh** at any page position including the top. This trap position may be changed using **.ch**. Trap positions at or below the bottom of the page have no effect unless or until moved to within the page or rendered effective by an increase in page length.

Two traps may be planted at the same position only by first planting them at different positions and then moving one of the traps; the first planted trap will conceal the second unless and until the first one is moved. If the first one is moved back, it again conceals the second trap.

The macro associated with a page trap is automatically invoked when a line of text is output whose vertical size reaches or sweeps past the trap position. Should several traps be placed so close together that a single output line could spring all of them, all but the first will be ignored. Reaching the bottom of a page springs the top-of-page trap, if any, provided there is a next page. The distance to the next trap position is available in the **.t** register; if there are no traps between the current position and the bottom of the page, the distance returned is the distance to the page bottom.

A macro-invocation trap effective in the current diversion may be planted using **.dt**. The **.t** register works in a diversion; if there is no subsequent trap a large distance is returned.

Request Form	Initial Value	If No Argument	Notes	Explanation
.de *xx yy*	-	*.yy*=..	-	Define or redefine the macro *xx*. The contents of the macro begin on the next input line. Input lines are copied in copy mode until the definition is terminated by a line beginning with *.yy*, whereupon the macro *yy* is called. In the absence of *yy*, the definition is terminated by a line beginning with "..". A macro may contain **.de** requests provided the terminating macros differ or the contained definition terminator is concealed; ".." can be concealed as "\\.." which will copy as "\.." and be reread as "..".
.am *xx yy*	-	*.yy*=..	-	Append to macro (append version of **.de**).

.ds *xx string*	ignored	-	-	Define a string *xx* containing *string*. Any initial double-quote in *string* is stripped off to permit initial blanks.
.as *xx string*	ignored	-	-	Append *string* to string *xx* (append version of **.ds**).
.rm *xx*	-	ignored	-	Remove request, macro, or string. The name *xx* is removed from the name list and any related storage space is freed. Subsequent references will have no effect.
.rn *xx yy*	-	ignored	-	Rename request, macro, or string *xx* to *yy*. If *yy* exists, it is first removed.
.di *xx*	-	end	D	Divert output to macro *xx*. Normal text processing occurs during diversion except that page offsetting is not done. The diversion ends when the request **.di** or **.da** is encountered without an argument; extraneous requests of this type should not appear when nested diversions are being used.
.da *xx*	-	end	D	Divert, appending to *xx* (append version of **.di**).
.wh *N xx*	-	-	v	Install a trap to invoke *xx* at page position *N;* a negative *N* will be interpreted with respect to the page bottom. Any macro previously planted at *N* is replaced by *xx*. A zero *N* refers to the top of a page. In the absence of *xx*, the first found trap at *N*, if any, is removed.
.ch *xx N*	-	-	v	Change the trap position for macro *xx* to be *N*. In the absence of *N*, the trap, if any, is removed.
.dt *N xx*	-	off	D,v	Install a diversion trap at position *N* in the current diversion to invoke macro *xx*. Another **.dt** will redefine the diversion

				trap. If no arguments are given, the diversion trap is removed.
.it *N xx*	-	off	E	Set an input-line-count trap to invoke the macro *xx* after *N* lines of text input have been read (control or request lines don't count). The text may be in-line text or either in-line or trap-invoked macros representing text. (See the discussion of the input-line-count **.it** request in section 7.5, "Traps.")
.em *xx*	none	none	-	The macro *xx* will be invoked when all input has ended. The effect is the same as if the contents of *xx* had been at the end of the last file processed.

8. Number Registers

A variety of parameters are available to the user as predefined, named number registers. The user may also define his own named registers. Register names are one or two characters long and do not conflict with request, macro, or string names. Except for certain predefined read-only registers, a number register can be read, written, automatically incremented or decremented, and inserted into the input using a variety of counting methods. Automatically numbered sections and paragraphs are common usages. A number register may be used any time numerical input is expected or desired. (See section 1.4, "Numerical Expressions.")

Number registers are created and modified using **.nr**, which specifies the name, numerical value, and the auto-increment size. Registers are also modified, if accessed with an auto-incrementing sequence. If the registers *x* and *xx* both contain *N* and have the auto-increment size *M*, the following access sequences have the effect shown:

Sequence	Effect on Register	Interpreted Value
\n*x*	none	*N*
\n(*xx*	none	*N*
\n+*x*	*x* incremented by *M*	*N+M*

\n−*x*	*x* decremented by *M*	*N*−*M*
\n+(*xx*	*xx* incremented by *M*	*N*+*M*
\n−(*xx*	*xx* decremented by *M*	*N*−*M*

When evaluated, a number register is converted to decimal (default), decimal with leading zeros, lower-case roman, upper-case roman, lower-case alphabetic, or upper-case alphabetic according to the format specified by **.af**. The escape sequence **\g***xx* or **\g(***xx* gives the format used by the registers *x* or *xx*. This escape sequence will only return a value if the stated register has been set or used; otherwise, it returns 0. The value can also be saved and used as the second argument to **.af** to restore a previous format.

Request Form	Initial Value	If No Argument	Notes	Explanation
.nr *R* ±*N M*	-	-	**u**	The number register *R* is assigned the value ±*N* with respect to the previous value, if any. The increment for auto-incrementing is set to *M*.
.af *R c*	Arabic	-	-	Assign format *c* to register *R*. The available formats are

Format	Numbering Sequence
1	0,1,2,3,4,5,...
001	000,001,002,003,004,005,...
i	0,i,ii,iii,iv,v,...
I	0,I,II,III,IV,V,...
a	0,a,b,c,...,z,aa,ab,...,zz,aaa,...
A	0,A,B,C,...,Z,AA,AB,...,ZZ, AAA,...

An Arabic format having *N* digits specifies a field width of *N* digits. The read-only registers and the width function are always Arabic. (See section 11.2, "Width Function.")

.rr *R*	-	ignored	-	Remove register *R*. If many registers are being created dynamically, it may become necessary to remove unneeded registers to

recapture internal storage space for new registers.

9. Tabs, Leaders, and Fields

9.1. Tabs and Leaders

The ASCII horizontal tab character and the ASCII SOH (hereafter known as the leader character) can both be used to generate either horizontal motion or a string of repeated characters. The length of the generated entity is governed by internal tab stops specifiable with **.ta**. The default difference is that tabs generate motion and leaders generate a string of periods; **.tc** and **.lc** offer the choice of repeated character or motion. There are three types of internal tab stops—left adjusting, right adjusting, and centering. In the following table *D* is the distance from the current position on the input line (where a tab or leader was found) to the next tab stop; next-string consists of the input characters following the tab (or leader) up to the next tab (or leader) or end of line; and *W* is the width of next-string.

Tab type	Length of motion or repeated characters	Location of next-string
Left	*D*	Following *D*
Right	*D*−*W*	Right adjusted within *D*
Centered	*D*−*W*/2	Centered on right end of *D*

The length of generated motion is allowed to be negative, but that of a repeated character string cannot be. Repeated character strings contain an integer number of characters, and any residual distance is prepended as motion. Tabs or leaders found after the last tab stop are ignored, but may be used as next-string terminators.

Tabs and leaders are not interpreted in copy mode. **\t** and **\a** always generate a non-interpreted tab and leader respectively, and are equivalent to actual tabs and leaders in copy mode but are ignored during output mode.

9.2. Fields

A field is contained between a pair of field delimiter characters, and consists of sub-strings separated by padding indicator characters. The field length is the distance on the input line from the position where the field begins to the next tab stop. The difference between the total length of all the sub-strings and the field length is incorporated as horizontal padding space that is divided among the indicated padding

places. The incorporated padding is allowed to be negative. For example, if the field delimiter is **#** and the padding indicator is ^, *#^xxx^right#* specifies a right-adjusted string with the string *xxx* centered in the remaining space.

Request Form	Initial Value	If No Argument	Notes	Explanation
.ta *Nt ...*	8n; 0.5in	none	E,**m**	Set tab stops and types. *t*=**R**, right adjusting; *t*=**C**, centering; *t* absent, left adjusting. **troff** tab stops are preset every 0.5in.; **nroff** every 8 nominal character widths. The stop values are separated by spaces, and a value preceded by **+** is treated as an increment to the previous stop value.
.tc *c*	none	none	E	The tab repetition character becomes *c*, or is removed specifying motion.
.lc *c*	.	none	E	The leader repetition character becomes *c*, or is removed specifying motion.
.fc *a b*	off	off	-	The field delimiter is set to *a*; the padding indicator is set to the space character or to *b*, if given. In the absence of arguments the field mechanism is turned off.

10. Input and Output Conventions and Character Translations

10.1. Input Character Translations

Ways of typing in the graphic character set are discussed in section 2.1 "Character Set." The ASCII control characters SOH and horizontal tab are described in section 9.1, "Tabs and Leaders." The backspace is discussed in section 10.3, "Backspacing, Underlining, and Overstriking." The new-line delimits input lines. In addition, STX, ETX, ENQ, ACK, and BEL may be used as delimiters or translated into a graphic with **.tr**. (See section 10.5, "Output Translation.") **troff** normally passes none of these characters to its output; **nroff** passes the BEL character. All others are ignored.

The escape character \ introduces escape sequences—causes the following character to mean another character, or to indicate some function. A complete list of such sequences is given in the "Summary" above. \ should not be confused with the ASCII control character ESC of the same name. The escape character \ can be input with the sequence \\. The escape character can be changed with **.ec**, and all that has been said about the default \ becomes true for the new escape character. **\e** can be used to print whatever the current escape character is. If necessary or convenient, the escape mechanism may be turned off with **.eo**, and restored with **.ec**.

Request Form	Initial Value	If No Argument	Notes	Explanation
.ec *c*	\	\	-	Set escape character to \, or to *c*, if given.
.eo	on	-	-	Turn escape mechanism off.

10.2. Ligatures

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

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

10.3. Backspacing, Underlining, and Overstriking

Unless in copy mode, the ASCII backspace character is replaced by a backward horizontal motion having the width of the space character. Underlining as a form of line-drawing is discussed in section 12.4, "Line Drawing." A generalized overstriking function is described in section 12.1, "Overstriking."

nroff automatically underlines characters in the underline font specifiable with **.uf** (normally Times Italic on font position 2.) See section 2.2, "Fonts." In addition to **.ft** and **\f***F*, the underline font is selected by **.ul** and **.cu**. Underlining is restricted to an output-device-dependent subset of reasonable characters.

Request Form	Initial Value	If No Argument	Notes	Explanation
.ul *N*	off	*N*=1	E	Underline in **nroff** (italicize in **troff**) the next *N* input text lines. Actually, switch to underline font, saving the current font for later restoration; other font changes within the span of a **.ul** will take effect, but the restoration will undo the last change. Output generated by **.tl** is affected by the font change, but does not decrement *N*. (See section 14, "Three Part Titles.") If *N*>1, there is the risk that a trap-invoked macro may provide text lines within the span; environment switching can prevent this.
.cu *N*	off	*N*=1	E	A variant of **.ul** that causes every character to be underlined and causes no line breaks to occur in the affected input

				lines. That is, each output space following **.cu** is like an unpaddable space. **.cu** is identical to **.ul** in **troff**.
.uf *F*	Italic	Italic	-	Underline font set to *F*. In **nroff**, *F* may not be on position 1 (initially Times Roman).

10.4. Control Characters

Both the control character "." and the no-break control character "´" may be changed, if desired. Changes must be compatible with the design of macros used in the span of the change, especially trap-invoked macros.

Request Form	Initial Value	If No Argument	Notes	Explanation
.cc *c*	.	.	E	The basic control character is set to *c*, or reset to ".".
.c2 *c*	´	´	E	The no-break control character is set to *c*, or reset to "´".

10.5. Output Translation

One character can be made a stand-in for another character using **.tr**. All text processing (e.g., character comparisons) takes place with the input (stand-in) character which appears to have the width of the final character. The graphic translation occurs at the moment of output (including diversions).

Request Form	Initial Value	If No Argument	Notes	Explanation
.tr *abcd....*	none	-	O	Translate *a* into *b*, *c* into *d*, etc. If an odd number of characters is given, the last one will be mapped into the space character. To be consistent, a particular translation must stay in effect from input to output time. To reset **.tr**, follow request with previous arguments given in duplicate. The example given at the start of this entry, for instance, would be turned off as follows: **.tr** *aacc*.

10.6. Transparent Throughput

An input line beginning with a **\!** is read in copy mode and transparently output (without the initial **\!**); the text processor is otherwise unaware of the line's presence. This mechanism may be used to pass control information to a post-processor or to embed control lines in a macro created by a diversion.

10.7. Comments and Concealed New-lines

An awkwardly long input line that must stay one line (e.g., a string definition, or no-filled text) can be split into many physical lines by ending all but the last one with the escape ****. The sequence ****(new-line) is always ignored—except in a comment. Comments may be embedded at the end of any line by prefacing them with **\"**. The new-line at the end of a comment cannot be concealed. A line beginning with **\"** will appear as a blank line and behave like **.sp 1**; a comment can be on a line by itself by beginning the line with **.\"**.

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

11.1. Local Motions

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

Vertical Motion	Effect in R	Effect in **nroff**	Horizontal Motion	Effect in **troff**	Effect in **nroff**
\v'*N***'**	Move distance *N*		**\h'***N***'**	Move distance *N*	
			****(space)	Unpaddable space-size space	
\u	½ em up	½ line up	**\0**	Digit-size space	
\d	½ em down	½ line down			
\r	1 em up	1 line up	**|**	1/6 em space	ignored
			\^	1/12 em space	ignored

As an example, E^2 could be generated by the sequence
`E\s-2\v'-0.4m'2\v'0.4m'\s+2`; it should be noted in this example that the 0.4 em vertical motions are at the smaller point size.

11.2. Width Function

The width function **\w'***string***'** generates the numerical width of *string* (in basic units). Size and font changes may be safely embedded in *string*, and will not affect the current environment. For example, `.ti -\w'1. 'u` could be used to temporarily indent leftward a distance equal to the size of the string "**1.** ".

The width function also sets three number registers. The registers **st** and **sb** are set respectively to the highest and lowest extent of *string* relative to the baseline; then, for example, the total height of the string is **\n(stu-\n(sbu**. In **troff** the number register **.ct** is set to a value between 0 and 3: 0 means that all of the characters in *string* were short lower case characters without descenders (like "e"); 1 means that at least one character has a descender (like "y"); 2 means that at least one character is

tall (like "H"); and 3 means that both tall characters and characters with descenders are present.

11.3. Mark Horizontal Place

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

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

12.1. Overstriking

Automatically centered overstriking of up to nine characters is provided by the overstrike function **\o'***string***'**. The characters in *string* are overprinted with centers aligned; the total width is that of the widest character. *string* should not contain local vertical motion. As examples, **\o'e\''** produces **e\o'\(mo\(sl'** produces ∉.

12.2. Zero-width Characters

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

12.3. Large Brackets

The Special Mathematical Font contains a number of bracket construction pieces (⎛ ⎜ ⎝) ⎞ ⎟ ⎠ { ⎨ ⎬ | ⎢ ⎥ ⎡ ⎤) that can be combined into various bracket styles. The function **\b'***string***'** may be used to pile up vertically the characters in *string* (the first character on top and the last at the bottom); the characters are vertically separated by 1 em and the total pile is centered 1/2 em above the current baseline (½ line in **nroff**). For example, `\b'\(lc\(lf'E\|\b'\(rc\(rf'\x'-0.5m'\x'0.5m'` produces ⎡E⎤.

12.4. Line Drawing

The function **\l'***Nc***'** will draw a string of repeated *c*'s towards the right for a distance *N*. (**\l** is ****(lower-case L). If *c* looks like a continuation of an expression for *N*, it may be insulated from *N* with a **\&**. If *c* is not specified, the _ (baseline rule) is used (underline character in **nroff**). If *N* is negative, a backward horizontal motion of size *N* is made before drawing the string. Any space resulting from *N*/(size of *c*) having a remainder is put at the beginning (left end) of the string. In the case of characters that are designed to be connected such as baseline-rule _, underrule _, and rooten ‾, the remainder space is covered by overlapping. If *N* is less than the width of *c*, a single *c* is centered on a distance *N*. As an example, a macro to underscore a string can be written

```
.de us
\\$1\l'|0\(ul'
..
```

or one to draw a box around a string

```
.de bx
\(br\|\\$1\|\(br\l'|0\(rn'\l'|0\(ul'
..
```

such that

```
.us "underlined words"
```

and

```
.bx "words in a box"
```

yield underlined words and [words in a box]. Notice that the text follows the request on the same input line and that multiple words are enclosed by double quotes. (Compare the **mm** font macros **.R**, **.I**, **.B**, **.RI**, **.RB**, and so forth.) The output cannot cross line boundaries. One way to ensure this is to use these macros in no-fill mode where input lines and output lines are identical.

The function **\L'***Nc***'** will draw a vertical line consisting of the (optional) character *c* stacked vertically apart 1 em (1 line in **nroff**), with the first two characters overlapped, if necessary, to form a continuous line. The default character is the box rule | (**\(br**); the other suitable character is the bold vertical | (**\(bv**). The line is begun without any initial motion relative to the current base line. A positive *N* specifies a line drawn downward and a negative *N* specifies a line drawn upward. After the line is drawn no compensating motions are made; the instantaneous baseline is at the end of the line.

The horizontal and vertical line drawing functions may be used in combination to produce large boxes. The zero-width box-rule and the ½-em wide underrule were designed to form corners when using 1-em vertical spacings. For example the macro

```
.de eb
.sp -1      \" compensate for next automatic base-line spacing
.nf         \" avoid possibly overflowing word buffer
\h'-.5n'\L'|\\nau-1'\l'\\n(.lu+1n\(ul'\L'-|\\nau+1\
'\l'|0u-.5n\(ul'    \" draw box
.fi
..
```

will draw a box around some text whose beginning vertical place was saved in number register *a* (e.g., using **.mk a**) as done for this paragraph.

In addition, **troff** provides drawing functions capable of drawing arcs and splines.

Request Form	Initial Value	If No Argument	Notes	Explanation
\D'l *dh dv'*	-	-	-	Draw a line from the current position by *dh, dv.*
\D'c *d'*	-	-	-	Draw a circle of diameter *d* with its left side at the current position.
\D'e *d1 d2'*	-	-	-	Draw an ellipse of diameters *d1* and *d2* with its left side at the current position.
\D'a *dh1 dv1 dh2 dv2'*	-	-	-	Draw a counterclockwise arc from the current position to *dh1+dh2, dv1+dv2,* with its center at *dh1, dv1* from the current position.
\D'~ *dh1 dv1 dh2 dv2...'*	-	-	-	Draw a B-spline from the current position by *dh1, dv1,* then by *dh2, dv2,* then ...

The current position after using these drawing functions is at the end of the drawn line, which for circles and ellipses is at the right side.

13. Hyphenation

The automatic hyphenation may be switched off and on. When switched on with **.hy**, several variants may be set. A hyphenation indicator character may be embedded in a word to specify desired hyphenation points, or may be prepended to suppress hyphenation. In addition, the user may specify a small exception word list.

Only words that consist of a central alphabetic string surrounded by (usually null) non-alphabetic strings are considered candidates for automatic hyphenation. Words that were input containing hyphens (minus), em-dashes (**\(em**), or hyphenation indicator characters—such as mother-in-law—are always subject to splitting after those characters, whether or not automatic hyphenation is on or off.

Request Form	Initial Value	If No Argument	Notes	Explanation
.nh	no hyphen.	-	E	Automatic hyphenation is turned off.
.hy *N*	off,*N*=0	on,*N*=1	E	Automatic hyphenation is turned on for $N \geq 1$, or off for $N = 0$. If $N = 2$, last lines (ones that will cause a trap) are not hyphenated. For $N = 4$ and 8, the last and first two characters respectively of a word are not split off. These values are additive; i.e., $N = 14$ will invoke all three restrictions.
.hc *c*	\%	\%	E	Hyphenation indicator character is set to *c* or to the default **\%**. The indicator does not appear in the output.
.hw *word1 ...*		ignored		Specify hyphenation points in words with embedded minus signs. Versions of a word with terminal *s* are implied (that is, *dig–it* implies *dig–its*). This list is examined initially and after each suffix stripping. The space available is small—about 128 characters.

14. Three Part Titles

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

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

15. Output Line Numbering

Automatic sequence numbering of output lines may be requested with **.nm**. When in effect, a three-digit, arabic number plus a digit-space is prepended to output text lines. The text lines are thus offset by four digit-spaces, and otherwise retain their line length. A reduction in line length may be desired to keep the right margin aligned with an earlier margin. Blank lines, other vertical spaces, and lines generated by **.tl** are not numbered. Numbering can be temporarily suspended with **.nn**, or with an **.nm** followed by a later **.nm +0**. In addition, a line number indent *I*, and the number-text separation *S* may be specified in digit-spaces. Further, it can be specified that only those line numbers that are multiples of some number *M* are to be printed (the others will appear as blank number fields).

Request Form	Initial Value	If No Argument	Notes	Explanation
.nm ±*N M S I*		off	E	Line number mode. If ±*N* is given, line numbering is turned on, and the next output line numbered is numbered ±*N*. Default values are *M*=1, *S*=1, and *I*=0. Parameters corresponding to missing arguments are unaffected; a non-numeric argument is considered missing. In the absence of all arguments, numbering is turned off; the next line number is preserved for possible further use in number register **ln**.
.nn *N*	-	*N*=1	E	The next *N* text output lines are not numbered.

As an example, the paragraph portions of this section are numbered with *M*=3: **.nm 1 3** was placed at the beginning; **.nm** was placed at the end of the first paragraph; and **.nm +0** was placed in front of this paragraph; and **.nm** finally placed at the end. Line lengths were also changed (by **\w'0000'u**) to keep the right side aligned. Another example is **.nm +5 5 x 3** which turns on numbering with the line number of the next line to be 5 greater than the last numbered line, with *M*=5, with spacing *S* untouched, and with the indent *I* set to 3.

16. Conditional Acceptance of Input

In the following, *c* is a one-character, built-in condition name, ! signifies not, *N* is a numerical expression, *string1* and *string2* are strings delimited by any non-blank, non-numeric character not in the strings, and *anything* represents what is conditionally accepted.

Request Form	Initial Value	If No Argument	Notes	Explanation
.if *c anything*		-	-	If condition *c* true, accept *anything* as input; in multi-line case use *\{anything\}*.
.if *!c anything*		-	-	If condition *c* false, accept *anything*.
.if *N anything*		-	u	If expression $N > 0$, accept *anything*.
.if *!N anything*		-	u	If expression $N \leq 0$, accept *anything*.
.if *'string1'string2' anything*			-	If *string1* identical to *string2*, accept *anything*.
.if *!'string1'string2' anything*			-	If *string1* not identical to *string2*, accept *anything*.
.ie *c anything*		-	u	If portion of if-else; all above forms (like **.if**).
.el *anything*	-	-	-	Else portion of if-else.

The built-in condition names are

Condition Name	True If
o	Current page number is odd
e	Current page number is even
t	Formatter is **troff**
n	Formatter is **nroff**

If the condition *c* is true, or if the number *N* is greater than zero, or if the strings compare identically (including motions and character size and font), *anything* is accepted as input. If a ! precedes the condition, number, or string comparison, the sense of the acceptance is reversed.

Any spaces between the condition and the beginning of *anything* are skipped over. The *anything* can be either a single input line (text, macro, or whatever) or a number of input lines. In the multi-line case, the first line must begin with a left delimiter **\{** and the last line must end with a right delimiter **\}**. The left delimiter must be followed by either a command or text:

```
.if 54>1 \{ ´sp 0.5i
```

Following this delimiter with a backslash (**\{**), escaping the newline, yields an equivalent arrangement.

The request **.ie** (if-else) is identical to **.if** except that the acceptance state is remembered. A subsequent and matching **.el** (else) request then uses the reverse sense of that state. `.ie` - **.el** pairs may be nested.

Some examples are

```
.if e .tl ´Even Page %´´´
```

which outputs a title if the page number is even; and

```
.ie \n%>1 \{\
´sp 0.5i
.tl ´Page %´´´
´sp |1.2i \}
.el .sp |2.5i
```

which treats page 1 differently from other pages.

17. Environment Switching

A number of the parameters that control the text processing are gathered together into an environment, which can be switched by the user. The environment parameters are those associated with requests noting E in their *Notes* column; in addition, partially collected lines and words are in the environment. Everything else is global; examples are page-oriented parameters, diversion-oriented parameters, number registers, and macro and string definitions. All environments are initialized with default parameter values.

Request Form	Initial Value	If No Argument	Notes	Explanation
.ev *N*	*N*=0	previous	-	Environment switched to environment $0 \leq N \leq 2$. Switching is done in push-down fashion so that restoring a previous environment must be done with **.ev** rather than specific reference.

18. Insertions from the Standard Input

The input can be temporarily switched to the system standard input with **.rd**, which will switch back when two new-lines in a row are found (the extra blank line is not used). This mechanism is intended for insertions in form-letter-like documentation. On the DG/UX system, the standard input can be the user's keyboard, a pipe, or a file.

Request Form	Initial Value	If No Argument	Notes	Explanation
.rd *prompt*	-	*prompt*=BEL	-	Read insertion from the standard input until two new-lines in a row are found. If the standard input is the user's keyboard, *prompt* (or a BEL) is written onto the user's terminal. **.rd** behaves like a macro, and arguments may be placed after *prompt*.
.ex	-	-	-	Exit from **nroff/troff**. Text processing is terminated exactly as if all input had ended.

If insertions are to be taken from the terminal keyboard while output is being printed on the terminal, the command-line option **−q** will turn off the echoing of keyboard input and prompt only with BEL. The regular input and insertion input cannot simultaneously come from the standard input.

As an example, multiple copies of a form letter may be prepared by entering the insertions for all the copies in one file to be used as the standard input, and causing the file containing the letter to reinvoke itself using **.nx**; the process would ultimately be ended by an **.ex** in the insertion file.

19. Input/Output File Switching

Request Form	Initial Value	If No Argument	Notes	Explanation
.so *file*	-	-	-	Switch source file. The top input (file reading) level is switched to *file*. When the new file ends, input is again taken from the original file; **.so**'s may be nested. Note that *file* should be preprocessed, if necessary, before being called by **.so**. **eqn**, **tbl**, **pic**, and **grap** will not reach through **.so**s to process an object file. Once a **.so** is encountered, the processing of *file* is immediate. Processing of the original file (e. g., a macro that is still active) is suspended.
.nx *file*	end-of-file	-	-	Next file is *file*. The current file is considered ended, and the input is immediately switched to *file*.
.cf *file*	-	-	-	Copy the contents of *file*, uninterpreted into **troff** output file at this point. Havoc ensues unless the motions in the file restore the current horizontal and vertical position.
.lf *N file*	-	-	-	Corrects **troff**'s idea of the current line number, *N*, and the current file, *file*, for use in error messages.
.pi *program*	-	-	-	Pipe output to *program*. This request must occur before any printing occurs. No arguments are transmitted to *program*.

20. Miscellaneous

Request Form	Initial Value	If No Argument	Notes	Explanation
.mc *c N*	-	off	E,m	Specifies that a margin character *c* appear a distance *N* to the right of the right margin after each non-empty text line (except those produced by **.tl**). If the output line is too long (as can happen in no-fill mode) the character will be appended to the line. If *N* is not given, the previous *N* is used; the initial *N* is 0.2 inches in **nroff** and 1 em in **troff**. The margin character used with this paragraph was a 12-point box-rule.
.tm *string*	-	new-line	-	After skipping initial blanks, *string* (rest of the line) is read in copy mode and written on the user's terminal.
.ig *yy*	-	*.yy*=..	-	Ignore input lines. **.ig** behaves exactly like **.de** except that the input is discarded. (See section 7, "Macros, Strings, Diversions, and Position Traps.") The input is read in copy mode, and any auto-incremented registers will be affected.

.pm *t*	-	all	-	Print macros. The names and sizes of all of the defined macros and strings are printed on the user's terminal; if *t* is given, only the total of the sizes is printed. The size is given in blocks of 128 characters.
.fl	-	-	B	Flush output buffer.
.ab *text*	-	Abort	-	Prints *text* on the diagnostic output (normally the terminal) and terminates without further processing. If *text* is missing, the message "User Abort" is printed. The output buffer is flushed. Used in interactive debugging to force output.
.sy *cmd args*	-	-	-	*cmd* is executed but its output is not captured at this point. The standard input for *cmd* is closed. Output for processing must be explicitly saved in an output file.

21. Output and Error Messages

The output from **.tm**, **.pm**, and the prompt from **.rd**, as well as various error messages are written onto the DG/UX system's standard error message output. The latter is different from the standard output, where **nroff** formatted output goes. By default, both are written onto the user's terminal, but they can be independently redirected.

Various error conditions may occur during the operation of **nroff** and **troff**. Certain less serious errors having only local impact do not cause processing to terminate. Two examples are "word overflow," caused by a word that is too large to fit into the word buffer (in fill mode), and "line overflow," caused by an output line that grew too large to fit in the line buffer; in both cases, a message is printed, the offending excess is discarded, and the affected word or line is marked at the point of truncation with a * in **nroff** and a ☜ in **troff**. The philosophy is to continue processing, if possible, on the grounds that output useful for debugging may be produced. If a serious error occurs, processing terminates, and an appropriate message is printed. Examples are the inability to create, read, or write files, and the exceeding of certain internal limits that make future output unlikely to be useful.

Special Characters

1. Input Names for ´, `, and – and for Non-ASCII Special Characters

Char	Input Name	Character Name	Char	Input Name	Character Name
"	´´	close quote	fi	\(fi	fi ligature
"	``	open quote	ff	\(ff	ff ligature
'	\(fm	foot mark	ff	\(fl	fl ligature
¢	\(ct	cent sign	ffi	\(Fi	ffi ligature
—	\(em	3/4 em dash	ffl	\(Fl	ffl ligature
–	\-	current font minus	¼	\(14	one-fourth
_	\(ru	rule	½	\(12	one half
-	\(hy	hyphen	¾	\(34	three-fourths
-	-	literal hyphen	†	\(dg	dagger
°	\(de	degree	‡	\(dd	double dagger
•	\(bu	bullet	®	\(rg	registered
□	\(sq	square	©	\(co	copyright
■	\(bx	box	™	\(tm	trademark

2. Non-ASCII Characters and Minus on the Standard Fonts

The upper-case Greek letter names followed by † are mapped into upper-case English letters in whatever font is mounted on font position one (default Times Roman). The special math plus, minus, and equals are provided to insulate the appearance of equations from the choice of standard fonts.

Char	Input Name	Character N
A	\(*A	Alpha†
B	\(*B	Beta†
Γ	\(*G	Gamma
Δ	\(*D	Delta
E	\(*E	Epsilon†
Z	\(*Z	Zeta†
H	\(*Y	Eta†
Θ	\(*H	Theta
I	\(*I	Iota†
K	\(*K	Kappa†
Λ	\(*L	Lambda
M	\(*M	Mu‡
N	\(*N	Nu†
Ξ	\(*C	Xi
O	\(*O	Omicron†
Π	\(*P	Pi
P	\(*R	Rho†
Σ	\(*S	Sigma
T	\(*T	Tau†
ϒ	\(*U	Upsilon
Φ	\(*F	Phi
X	\(*X	Chi†
Ψ	\(*Q	Psi
Ω	\(*W	Omega

Char	Input Name	Character Name
α	\(*a	alpha
β	\(*b	beta
γ	\(*g	gamma
δ	\(*d	delta
ϵ	\(*e	epsilon
ς	\(*z	zeta
η	\(*y	eta
θ	\(*h	theta
ι	\(*i	iota
κ	\(*k	kappa
λ	\(*l	lambda
μ	\(*m	mu
ν	\(*n	nu
ξ	\(*c	xi
o	\(*o	omicron
π	\(*p	pi
ρ	\(*r	rho
σ	\(*s	sigma
ς	\(ts	terminal sigma
τ	\(*t	tau
υ	\(*u	upsilon
ϕ	\(*f	phi
χ	\(*x	chi
ψ	\(*q	psi
ω	\(*w	omega

Char	Input Name	Character Name	Char	Input Name	Character Name
+	\(pl	math plus	∞	\(if	infinity
−	\(mi	math minus	∂	\(pd	partial derivative
=	\(eq	math equals	∇	\(gr	gradient
*	\(**	math star	¬	\(no	not
´	\(aa	acute accent	∝	\(pt	proportional to
`	\(ga	grave accent	∅	\(es	empty set
_	\(ul	underrule	∈	\(mo	member of
/	\(sl	slash	|	\(or	or
		(matching backslash)	|	\(br	box vertical rule
√	\(sr	square root	☞	\(rh	right hand
	\(rn	root en extender	☜	\(lh	left hand
≥	\(>=	>=	○	\(ci	circle
≤	\(<=	<=	⎧	\(lt	left top of big curly bracket
≡	\(==	identically equal			
≃	\(|=	approx =	⎩	\(lb	left bottom
~	\(ap	approximates	⎫	\(rt	right top
≠	\(!=	not equal	⎭	\(rb	right bottom
→	\(->	right arrow	⎨	\(lk	left center of big curly bracket
←	\(<-	left arrow			
↑	\(ua	up arrow	⎬	\(rk	right center of big curly bracket
↓	\(da	down arrow			
×	\(mu	multiply	⎪	\(bv	bold vertical
÷	\(di	divide	⎣	\(lf	left floor (left bottom of big square bracket)
±	\(+-	plus-minus			
∪	\(cu	cup (union)	⎦	\(rf	right floor (right bottom)
∩	\(ca	cap (intersection)	⎡	\(lc	left ceiling (left top)
⊂	\(sb	subset of	⎤	\(rc	right ceiling (right top)
⊃	\(sp	superset of		\cs	control-shift indicator
⊆	\(ib	improper subset		\vs	visible space indicator
⊇	\(ip	improper superset	§	\(sc	section

Documenter's
Tool Kit
Technical
Summary
to the
DG/UX™ System

069-701041-00

Cut here and insert in binder spine pocket

DataGeneral
Data General Corporation, Westboro, Massachusetts 01580

069-701041-00