sun®
microsystems

Formatting Documents

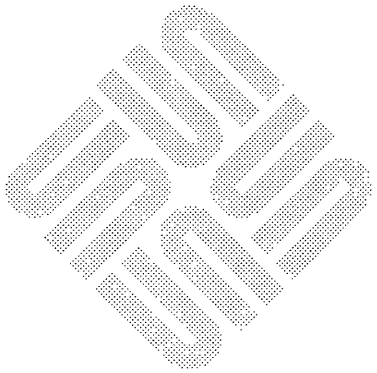
Part Number: 800-1756-10
Revision A, of 9 May 1988

Material in this manual comes from a number of sources: *Typing Documents on the UNIX System: Using the –ms Macros with Troff and Nroff,* M. E. Lesk, Bell Laboratories, Murray Hill, New Jersey; *A Guide to Preparing Documents with –ms,* M. E. Lesk, Bell Laboratories, Murray Hill, New Jersey; *Document Formatting on UNIX Using the –ms Macros,* Joel Kies, University of California, Berkeley, California; *Tbl — A Program to Format Tables,* M. E. Lesk, Bell Laboratories, Murray Hill, New Jersey; *A System for Typesetting Mathematics,* Brian W. Kernighan, Lorinda L. Cherry, Bell Laboratories, Murray Hill, New Jersey; *Typesetting Mathematics — User's Guide,* Brian W. Kernighan, Lorinda L. Cherry, Bell Laboratories, Murray Hill, New Jersey; *Updating Publications Lists,* M. E. Lesk, Bell Laboratories, Murray Hill, New Jersey; *Some Applications of Inverted Indexes on the UNIX System,* M. E. Lesk, Bell Laboratories, Murray Hill, New Jersey; *Writing Papers with Nroff Using –me,* Eric P. Allman, University of California, Berkeley; *The –me Reference Manual,* Eric P. Allman, University of California, Berkeley; and *Introducing the UNIX System,* Henry McGilton, Rachel Morgan, McGraw-Hill Book Company, 1983. These materials are gratefully acknowledged.

Contents

Tables

Figures



Preface

This manual provides user's guides and reference information for various document processing tools. We assume you are familiar with a terminal keyboard and the Sun system. If you are not, see *Getting Started with SunOS: Beginner's Guide* for information on the basics, like logging in and the Sun file system. If you are not familiar with text editing, read "An Introduction to Text Editing" in the manual *Editing Text Files*, or "An Introduction to Document Preparation" in this manual. Finally, we assume that you are using a Sun Workstation, although specific terminal information is also provided.

If you choose to read one of the user's guides, sit down at your workstation and try the exercises and examples. The reference sections provide additional explanations and examples on how to use certain facilities and can be dipped into as necessary. For additional details on Sun system commands and programs, see the *SunOS Reference Manual.*

Summary of Contents

This manual is divided into three sections:

- Macro Packages
- `troff` Preprocessors
- Verification and Reformatting Programs

1. *Introduction to Document Preparation* — Describes the basics of text processing, macros and macro packages, provides a guide to the available tools and several simple examples after which to pattern your papers and documents. Newcomers to the Sun document formatters should start here.

In Section I, Macro Packages, the chapters are:

2. *Formatting Documents with the* **-ms** *Macros* — User's guide and reference information for the **-ms** macros for formatting papers and documents. Includes new **-ms** macros.
3. *The* **-man** *Macro Package* — User's guide and reference information for the **-man** macros for formatting manual pages (*man* pages). Includes new options to the **-man** macro package.
4. *Formatting Documents with the* **-me** *Macros* — Describes the **-me** macro package for producing papers and documents.

In Section II, `troff` Preprocessors, the chapters are:

5. `refer` — *a Bibliography System* — Explains how to use the bibliographic citation program `refer`. Includes information on the auxiliary programs `addbib`, `indxbib`, `lookbib`, and `sortbib`.
6. *Formatting Tables with* `tbl` — A user's guide and numerous examples to the table processing utility `tbl`.
7. *Typesetting Mathematics with* `eqn` — A user's guide to the `eqn` mathematical equation processor.

Section III, Verification and Formatting Programs, discusses:

8. `checknr` — a program to report unmatched pairs of macros and unpaired font or size changes.

   `spell` — a program that prints strings of characters to your terminal screen that `spell` doesn't have in its dictionary (`/usr/dict/words`).

   The reformatting commands `fmt`, `deroff`, `pti`, `colcrt`, `col`, `ul`, and `ptx`.

Conventions Used in This Manual

Throughout this manual we use

```
hostname%
```

as the prompt to which you type system commands. **`Boldface typewriter font`** indicates commands that you type in exactly as printed on the page of this manual. `Regular typewriter font` represents what the system prints out to your screen. Typewriter font also specifies Sun system command names (program names) and illustrates source code listings. *Italics* indicates general arguments or parameters that you should replace with a specific word or string. We also occasionally use italics to emphasize important terms.

1

Introduction to Document Preparation

1

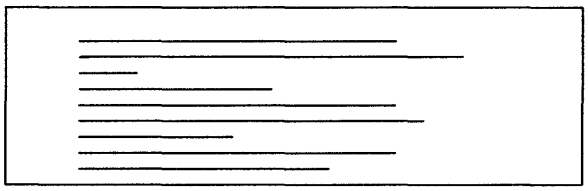
Introduction to Document Preparation

The document preparation tools `nroff` and `troff` are standard with SunOS. These programs read files containing the text to be formatted, interspersed with requests specifying how output should look. From this, the programs produce formatted output. `nroff` is for typewriter-like printers, while `troff` is for typesetters and laser printers. Although they are separate programs, they are compatible: the formatters share a common command language and produce output from the same input file. Descriptions here apply to both formatters unless stated otherwise.

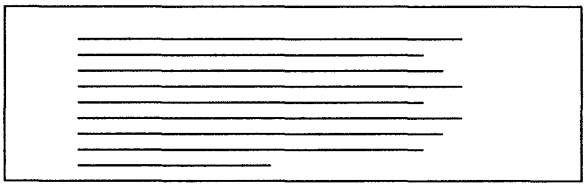
1.1. What Do Text Formatters Do?

You can type in text on lines of any length, and the formatters produce lines of uniform length in the finished document. This process is called *filling*, which means that the formatter collects words from what you type as input, and places them on an output line until no more fit within a given line length. The formatter *hyphenates* words automatically, so a line may end with part of a word to produce the right line length. The formatter also *adjusts* a line after it has been filled by inserting spaces between words as necessary to align the right margin exactly.

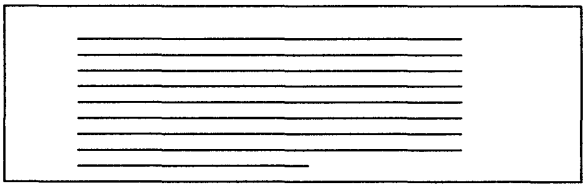
Unfilled text:



Filled but not adjusted:



Filled and adjusted:



Given a file of input consisting only of lines of text without any formatting requests, the formatter simply produces a continuous stream of filled, adjusted and hyphenated output.

To obtain paragraphs, numbered sections, multiple column layout, tops and bottoms of pages, and footnotes, for example, require the addition of formatting requests. Requests look like *.xx* where *xx* is one or two lower-case letters or a lower-case letter and a digit. Refer to *Using* `nroff` *and* `troff for details.`

1.2. What is a Macro Package?

`nroff` and `troff` provide a flexible, sophisticated command language for requesting operations like those just mentioned. They are very flexible, but this flexibility can make them difficult to use because you have to use several requests to produce a simple format. For this reason, it's a good idea to use a macro package.

A macro is simply a "predefined sequence of `troff` requests or text" which you can use by including just one request in your input file. You can then handle repetitious tasks, such as starting paragraphs and numbering pages, by typing one macro request each time instead of several. For example, some macro requests look like *.XX* where *XX* is one or two upper-case letters or an upper-case letter and a digit. (Different macro packages follow various conventions.)

A macro package also does a lot of things without the instructions that you have to give `nroff`, footnotes and page transitions for example. Some packages set up a page layout style by default, but you can change that style if you wish. Although a macro package offers only a limited subset of the wide range of formatting possibilities that `nroff` provides, it is much easier to use. We explain how to use a macro package in conjunction with `nroff` and `troff` in the section "Displaying and Printing Documents."

Sample input with both formatting requests, macros in this case, and text looks like:

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

Refer to the chapter "Formatting Documents with the **-ms** Macros" and to the "Quick References" in this chapter for more information on macros.

1.3. What is a Preprocessor?

A preprocessor is a program that you run your text file through first before passing it on to a text formatter. You can put tables in a document by preprocessing a file with the table-builder called `tbl`. You can add mathematical equations with their special fonts and symbols with the equation formatters, `eqn` for `troff` files and `neqn` for `nroff` files. These preprocessors convert material entered in their specific command languages to straight `troff` or `nroff` input. Those text formatters then produce the tables or mathematical equations for the output.

What you type in a file is very much the same as for simple formatting. You include table or equation material in your `troff` input file along with ordinary text and add several specific `tbl` or `eqn` requests. Refer to the chapters "Formatting Tables with `tbl`" and "Formatting Mathematics with `eqn`" for details.

1.4. Typesetting Jargon

There are several printer's measurement terms that are borrowed from traditional typesetting. These terms describe the size of the letters, the distance between lines and paragraphs, how long each line is, where the text is placed on the page, and so on.

Point — *Points* specify the *size* of a letter or *type*. A point measures about 1/72 of an inch, which means that there are 72 points to the inch. This manual is in 10-point type, for instance.

Ems and *Ens*

Ems and *ens* are measures of distance and are proportional to the type size being used. An *em* is the distance equal to the number of points in the width of the letter 'm' in that point size. For examples, here's an em in several point sizes followed by an em dash to show why this is a *proportional* unit of measure. You wouldn't want a 20-point dash if you are printing the rest of a document in 12-point. Here's 12-point:

And here's 20-point:

An *en* space is one half of an *em* or about the width of the letter 'n'. Ens are typically used for indicating indentation.

Vertical Spacing

Vertical spacing called *leading* (pronounced 'led-ding') is the distance between the bottom of one line and the bottom of the next. This manual has 12-point vertical spacing for example. The rule of thumb is that the spacing be approximately 20% larger than the character size for easy readability. A printer would call the ratio for this manual "ten on twelve."

Paragraph Depth

As there is a specification for the distance between lines, there is also a term for the space between paragraphs. This is the paragraph depth. If you are using the standard `.PP` or `.LP` macro, for instance, the paragraph depth is whatever one vertical space has been set to.

Paragraph Indent

This is the amount of space that the first line is indented in relation to the rest of the paragraph. If you use a `.PP` macro to format a standard indented

paragraph, the indent is two em-spaces as shown by the first line in this paragraph.

Line Length

Line length specifies the width of text on a page. Here we use a 5-inch line length. Shortening the line length generally makes text easier to read. Recall that many magazines and newspapers have 2-1/4 inch columns for quick reading.

Page Offset

Page offset determines the left margin, that is how far in the text is set from the left edge of the paper. On a normal 8-1/2-by-11 letter-size page, the page offset is normally 26/27 of an inch.

Indent The *indent* of text is the distance the text is set in from the page offset. The indent emphasizes the text by setting it off from the rest.

1.5. Hints for Typing in Text

The following provides a few tricks for typing in text and for further online editing and formatting.

- A period (`.`) or apostrophe (`'`) as the first character on a line indicates that the line contains a formatting request. If you type a line of text beginning with either of these *control characters*, `nroff` tries to interpret them as a request, and the rest of the text on that line disappears. If you *have* to type a period or an apostrophe as the first character on a line, escape their normal meanings by prefixing them with a backslash and an ampersand. For instance, to display this sample input:

```
\&.LP
Here is some sample input for a left-blocked paragraph.  In order
to accurately display -ms or troff requests that
begin lines, you have to precede them with the character sequence
backslash, ampersand (\&).  This insulates the macro request
from the beginning of the line so the dot in the first column isn't
seen by troff.
\&.LP
\&.sp
The .LP, .EQ and .EN requests shown here are -ms macro requests
and the .sp line is a typical troff request.
\&.EQ (1.3)
x sup 2 over a sup 2 ~=~ sqrt {p z sup 2 +qz+r}
\&.EN
```

- Following the control character is a one- or two-character *name* of a formatting request. As described earlier, `nroff` and `troff` names usually consist of one or two lower-case letters or a lower-case letter and a digit. **-ms** macro package names usually consist of one or two upper-case letters or one upper-case letter and a digit. For example, `.sp` is an `troff` request for a space and `.PP` is an **-ms** macro request for an indented paragraph.
- End a line of text with the end of a word along with any trailing punctuation. `nroff` inserts a space between whatever ends one line of input text and whatever begins the next.

- Start lines in the input file with something other than a space. A space at the beginning of an input line creates a *break* at that point in the output and `nroff` skips to a new output line, interrupting the process of filling and adjusting. This is the easiest way to get spaces between paragraphs, but it does not leave much flexibility for changing things later.
- Some requests go on a line by themselves, while others can take one or more additional pieces of information on the same line. These extra pieces of information on the request line are called *arguments*. Separate them from the request name and from each other by one or more spaces. Sometimes the argument is a piece of text on which the request operates; other times it can be some additional information about what the request is to do. For example, the vertical space request `.sp 3` shows a `troff` request with one argument. It requests three blank lines.

1.6. Types of Paragraphs

There are several types of paragraphs. When should you use one type of paragraph instead of another? Here are a few words about paragraphs, their characteristics, and formatting in general. See the "Types of Paragraphs" figure that follows for examples.

Use regular indented and block paragraphs for narrative descriptions. It is a matter of style as to which type you choose to use. In general, indented paragraphs remove the need for extra space between paragraphs — the indent tells you where the start of the new paragraph is. Most business communication is done with block paragraphs.

If you want to indicate a set of points without any specific order, use a bulleted list. For example:

There are many kinds of coffee:

- Jamaica Blue Mountain
- Colombian
- Java
- Mocha
- French Roast
- Major Dickenson's Blend

When you want to describe a set of things in some order, such as a step-by-step procedure, use a numbered list:

To repair television, follow these steps:

1. Remove screws in rear casing.
2. Carefully slide out picture tube.
3. Gently smash with hammer.

Use description lists to explain a set of related or unrelated things, or sometimes to highlight keywords. For instance,

Options

–v	Verbose
–f *filename*	Take script from *filename*
–o	Use old format

In typographic parlance, anything that is not part of the "body text" — regular paragraphs and such — is considered a *display*, and often has to be specially handled. Generally a display is "displayed" exactly as you type it or draw it originally, with no interference from the formatter. Displays are used to set off important text, special effects, drawings, or examples, as we do throughout this manual, The following paragraph is a *display*:

> Tom appeared on the sidewalk with a bucket of whitewash and
> a long-handled brush.
> He surveyed the fence, and all gladness left him and
> a deep melancholy settled down upon his spirit.
> Thirty yards of board fence nine feet high.
> Life to him seemed hollow, and
> existence but a burden.

Quotations set off quoted material from the rest of the text for emphasis. For example,

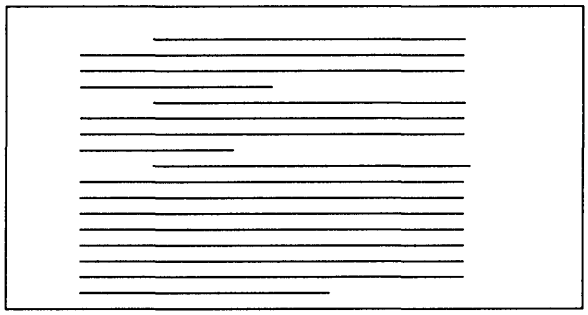
"... in the conversation between Alice and the Queen, we read this piece of homespun philosophy:

> "A slow sort of country!" said the Queen. "Now, *here*, you see, it takes all the running you can do, to keep in the same place. If you want to get somewhere else, you must run at least twice as fast as that!"
>
> *Through the Looking Glass*
> Lewis Carroll

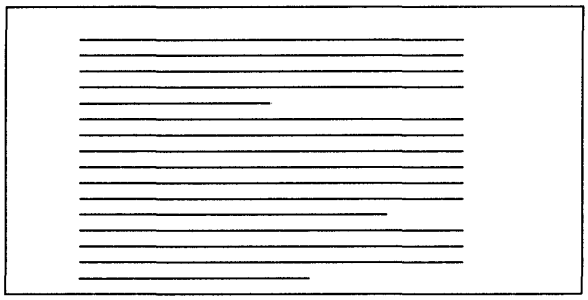
Paragraph Illustrations

Examine this section to see how the various paragraph types can serve different functions.

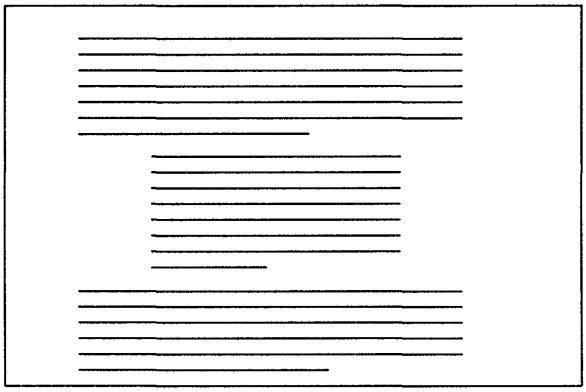
Indented — `.PP`



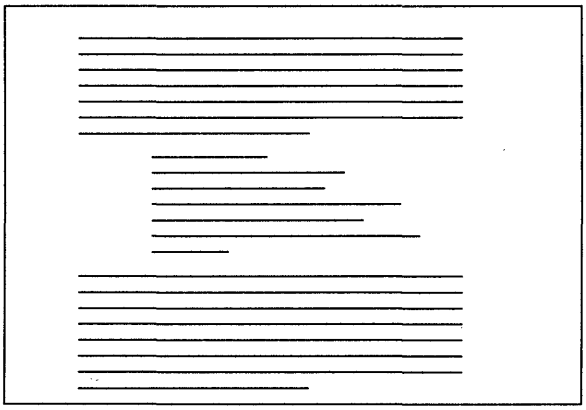
Left Block — `.LP`



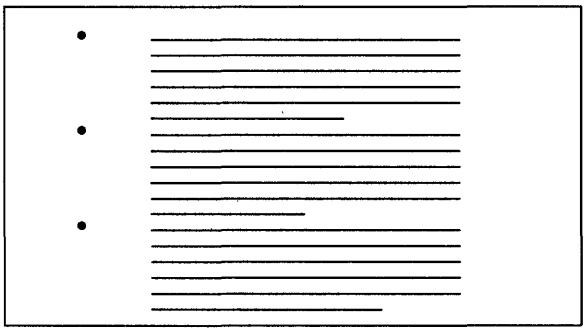
Quotation — `.QP`



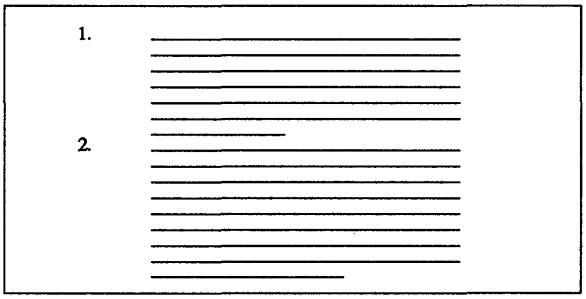
Display — `.DS`



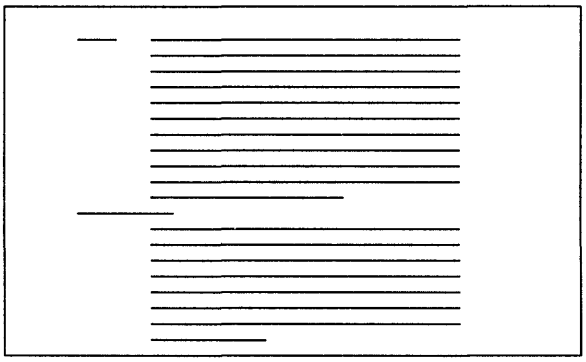
Bulleted — `.IP \(bu`



Numbered — `.IP 1.`



Lists — `.IP "tag" n`



1.7. Quick References

This section provides some simple templates for producing your documents with the **-ms** macro package.[1] Remember that for a quick, paginated, and justified document, you can simply type an `.LP` to start your document, and then type in the text separated by blank lines to produce paragraphs. Throughout the examples, the text file input is displayed in

```
typewriter font like this
```

while the output is displayed in

Times Roman font.

[1] Some of the material in this section is derived from *A Guide to Preparing Documents with '–ms'*, M.E. Lesk, Bell Laboratories, Murray Hill, New Jersey.

Displaying and Printing Documents

Use the following to format and print your documents. You can use either `nroff` or `troff` depending on the output you desire. Use `nroff` to either display formatted output on your workstation screen or to print a formatted document. The default is to display on the standard output, your workstation screen. For easy viewing, pipe your output to `more` or redirect the output to a file.

Using `troff` or your installation's equivalent prepares your output for phototypesetting.

Table 1-1 *How to Display and Print Documents*

What You Want to Do	*How to Do It*
Display simple text	**nroff** *–options files*
Display text with tables only	**tbl** *files* **| nroff** *–options*
Display text with equations only	**neqn** *files* **| nroff** *–options*
Display text with both tables and equations	**tbl** *files* **| neqn | nroff** *–options*
Print raw text and requests	**pr** *files* **| lpr** *–Pprinter*
Print text	**nroff** *–options files* **| lpr** *–Pprinter*
Print text with tables only	**tbl** *files* **| nroff** *–options* **| lpr** *–Pprinter*
Print text with equations only	**neqn** *files* **| nroff** *–options* **| lpr** *–Pprinter*
Print text with both tables and equations	**tbl** *files* **| neqn | nroff** *–options* **| lpr** *–Pprinter*
Phototypeset simple text	**troff** *–options files*
Phototypeset text with tables	**tbl** *files* **| troff** *–options*
Phototypeset text with equations	**eqn** *files* **| troff** *–options*
Phototypeset text with both tables and equations	**tbl** *files* **| eqn | troff** *–options*

Technical Memorandum

Here we provide a sample format for a technical memorandum.

Input:

```
.DA March 11, 1983
.TL
An Analysis of
Cucumbers and Pickles
.AU
A. B. Hacker
.AU
C. D. Wizard
.AI
Stanford University
Stanford, California
.AB
This abstract should be short enough to
fit on a single page cover sheet.
It provides a summary of memorandum
contents.
.AE
.NH
Introduction.
.PP
Now the first paragraph of actual text ...
...
Last line of text.
.NH
References
```

Output:

An Analysis of
Cucumbers and Pickles

A. B. Hacker
C. D. Wizard

Stanford University
Stanford, California

ABSTRACT

This abstract should be short enough to fit on a single page cover sheet. It provides a summary of memorandum contents.

1. Introduction.

Now the first paragraph of actual text ...
Last line of text.

2. References

Section Headings for Documents

```
.NH                          .SH
Introduction.                Appendix I
.PP                          .PP
text text text               text text text

1. Introduction              Appendix I
   text text text               text text text
```

Changing Fonts

The following table shows the easiest way to change the default roman font to italic or bold. To change the font of a single word, put the word on the same line as the macro request. To change the font in more than one word, put the text on the lines following the macro request.

The font will remain changed until another font change request or a macro request causing a break (a paragraph macro, for example) is encountered.

Input	Output
`.I Hello`	*Hello*
`.I` `Prints this line in italics.`	*Prints this line in italics.*
`.B Goodbye`	**Goodbye**
`.B` `Prints this line in bold.`	**Prints this line in bold.**
`.R` `Prints this line in roman.`	Prints this line in roman.

Making a Simple List

Use the following template for a simple list.

Input:

```
.IP 1.
J. Pencilpusher and X. Hardwired,
.I
A New Kind of Set Screw,
.R
Proc. IEEE
.B 75
(1976), 23-255.
.IP 2.
H. Nails and R. Irons,
.I
Fasteners for Printed Circuit Boards,
.R
Proc. ASME
.B 23
(1974), 23-24.
.LP        (terminates list)
```

Output:

1. J. Pencilpusher and X. Hardwired, *A New Kind of Set Screw*, Proc. IEEE **75** (1976), 23-255.
2. H. Nails and R. Irons, *Fasteners for Printed Circuit Boards*, Proc. ASME **23** (1974), 23-24.

Multiple Indents for Lists and Outlines

This template shows how to format lists or outlines.

Input:

```
This is ordinary text to highlight the
results of outline format.
.IP 1.
First level item.
.RS
.IP a)
Second level.
.IP b)
Continued here with another second
level item, but somewhat longer.
.RE
.IP 2.
Return to previous value of the
indenting at this point.
.IP 3.
Another
line.
```

Output:

This is ordinary text to highlight the results of outline format.

1. First level item.
    a) Second level.
    b) Continued here with another second level item, but somewhat longer.
2. Return to previous value of the indenting at this point.
3. Another line.

Displays

A display does not fill or justify the text. It keeps the text together, and sets the lines off from the rest.

Input:

```
hoboken harrison newark roseville avenue grove street
east orange brick church orange highland avenue
mountain station south orange maplewood millburn short hills
summit new providence
.DS
and now
for something
completely different
.DE
murray hill berkeley heights
gillette stirling millington lyons basking ridge
bernardsville far hills peapack gladstone
```

Output:

hoboken harrison newark roseville avenue grove street east orange brick church orange highland avenue mountain station south orange maplewood millburn short hills summit new providence

and now
for something
completely different

murray hill berkeley heights gillette stirling millington lyons basking ridge bernardsville far hills peapack gladstone

Display Options	Description
.DS L	left-adjust
.DS C	line-by-line center
.DS B	make block, then center

Footnotes

For automatically-numbered footnotes, put the string ** at the end of the text you want to footnote like this:[2]

```
you want to footnote like this:\**
.FS
Here's a numbered footnote.
.FE
```

To mark footnotes with other symbols, put the symbol as the first argument to .FS and at the end of the text you want to footnote like this:†

[2] Here's a numbered footnote.

† You can also use an asterisk (*) or a double dagger ‡ (\(dd).

```
you want to footnote like this:\(dg
.FS \(dg
You can also use an asterisk (\fL*\fR)
or a double dagger ‡ (\fL\(dd\fR).
.FE
```

Keeping Text Together — Keeps

Lines bracketed by the following commands are kept together, and will appear entirely on one page:

`.KS`		`.KF`	
lines of text		*lines of text*	
lines of text	not moved	*lines of text*	may float
lines of text	through text	*lines of text*	in text
lines of text		*lines of text*	
`.KE`		`.KE`	

Double-Column Format

Put a `.2C` at the beginning of the material you want printed in two columns. To return to one-column format, use `.1C`. Note that `.1C` breaks to a new page.

Input:

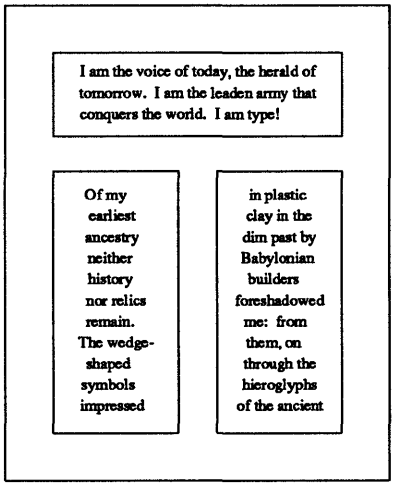
```
.TL
The Declaration of Independence
.sp 2
.2C
.LP
When in the course of human events, it becomes necessary
for one people to dissolve the political bonds which have
connected them with another, and to assume among the
powers of the earth the separate and equal station to which
the laws of Nature and of Nature's God entitle them,
a decent respect to the opinions of . . .
```

Output:

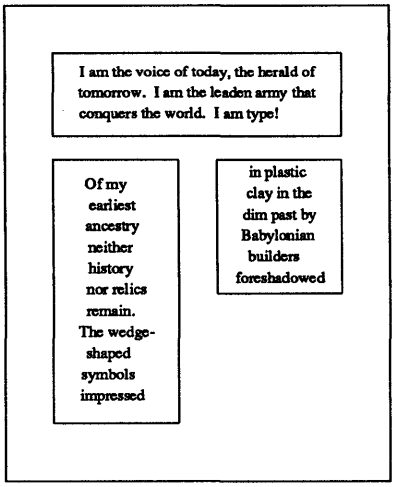
The Declaration of Independence

When in the course of human events, it becomes necessary for one people to dissolve the political bonds which have connected them with another, and to assume among the powers of the earth the separate and equal station to which the laws of Nature and of Nature's God entitle them, a decent respect to the opinions of . . .

The .2C macro request only works in this way: When you invoke the .2C macro somewhere on a page of text, .2C marks that height on the page and produces a narrow column of text all the way to the bottom of that page. When it reaches the bottom of the page, .2C resumes the second column at the height it originally marked off when the .2C macro was invoked. If the second column is only partially filled up with text when the .1C request is encountered, a page break occurs and the single-column text begins the next page. This means .2C will do this:

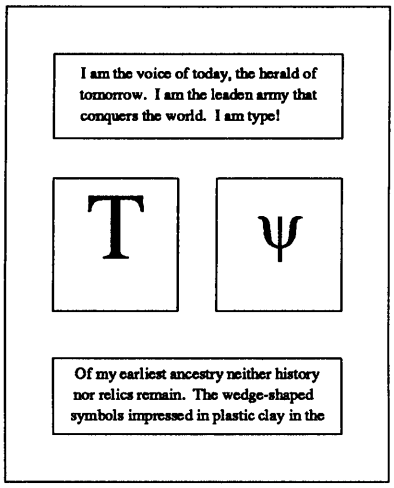


or this:



The important fact to remember about this macro request, is that it has no other way to determine where to begin column two except upon reaching the bottom of the page.

If the material you want in two columns occupies less space than the distance to the bottom of the current page, it will only occupy one narrow column if you use the .2C macro request. This means you cannot do this:



Sample Tables

Two sample table templates follow.

Input:

```
.TS
box center tab (/);
lB lB
l l.
Column Header   Column Header
_
text/text
text/text
text/text
text/text
.TE
```

Output:

Column Header	Column Header
text	text
text	text
text	text
text	text

Input:

```
.TS
allbox tab (/);
cB s s
c c c
n n n.
AT&T Common Stock
Year/Price/Dividend
1971/41-54/$2.60
2/41-54/2.70
3/46-55/2.87
4/40-53/3.24
5/45-52/3.40
6/51-59/.95*
.TE
* (first quarter only)
```

Output:

AT&T Common Stock		
Year	Price	Dividend
1971	41-54	$2.60
2	41-54	2.70
3	46-55	2.87
4	40-53	3.24
5	45-52	3.40
6	51-59	.95*

* (first quarter only)

The meanings of the key-letters describing the alignment of each entry are:

`tbl` Key-Letter	Column Described
`c`	centered
`r`	right-adjusted
`l`	left-adjusted
`n`	numerical
`a`	alphabetical
`s`	spanned

The global table options are `center`, `expand`, `box`, `doublebox`, `allbox`, `tab` (*x*), and `linesize` (*n*).

Input:

```
.TS
center box tab (/) ;
cB cB
l l.
Name/Definition
_
Gamma/$GAMMA (z) = int sub 0 sup inf t sup {z-1} e sup -t dt$
Sine/$sin (x) = 1 over 2i ( e sup ix - e sup -ix )$
Error/$roman erf (z) = 2 over sqrt pi int sub 0 sup z e sup {-t sup 2} dt$
Bessel/$J sub 0 (z) = 1 over pi int sub 0 sup pi cos ( z sin theta ) d theta$
Zeta/$zeta (s) = sum from k=1 to inf k sup -s ~~( Re~s > 1)$
.TE
```

Output:

Name	Definition
Gamma	$\Gamma(z)=\int_0^\infty t^{z-1}e^{-t}\,dt$
Sine	$\sin(x)=\frac{1}{2i}(e^{ix}-e^{-ix})$
Error	$\mathrm{erf}(z)=\frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}dt$
Bessel	$J_0(z)=\frac{1}{\pi}\int_0^\pi \cos(z\sin\theta)d\theta$
Zeta	$\zeta(s)=\sum_{k=1}^{\infty}k^{-s}$ (Re $s>1$)

Writing Mathematical Equations

A displayed equation is marked with an equation number at the right margin by adding an argument to the `.EQ` line:

Input:

```
.EQ (1.3)
x sup 2 over a sup 2 ~=~ sqrt {p z sup 2 +qz+r}
.EN
```

A displayed equation is marked with an equation number at the right margin by adding an argument to the `.EQ` line:

Output:

$$\frac{x^2}{a^2}=\sqrt{pz^2+qz+r} \qquad (1.3)$$

Input:

```
.EQ (2.2a)
bold V bar sub nu~=~left [ pile {a above b above
c } right ] + left [ matrix { col { A(11) above .
above . } col { . above . above .} col {. above .
above A(33) }} right ] cdot left [ pile { alpha
above beta above  gamma } right ]
.EN
```

Output:

$$\overline{\mathbf{V}}_{\nu} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} + \begin{bmatrix} A(11) & . & . \\ . & . & . \\ . & . & A(33) \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \qquad (2.2a)$$

Input:

```
.EQ I "" 2.75i
F hat ( chi ) ~ mark = ~ | del V | sup 2
.EN
.EQ I "" 2.75i
lineup =~ {left ( {partial V} over {partial x} right ) } sup 2
+ { left ( {partial V} over {partial y} right ) } sup 2
~~~~~~ lambda -> inf
.EN
```

Output:

$$\hat{F}(\chi) = |\nabla V|^2$$

$$= \left[\frac{\partial V}{\partial x}\right]^2 + \left[\frac{\partial V}{\partial y}\right]^2 \qquad \lambda \rightarrow \infty$$

Input:

```
$ a dot $,  $ b dotdot$,  $ rho tilde~times~y vec$.
```

Output:

$\dot{a}$, $\ddot{b}$, $\tilde{\rho} \times \vec{y}$.
(with delim $$ on).

Registers You Can Change

Table 1-2 *Registers You Can Change*

Register Name	Controls	Default Setting	Command to Change	Takes Effect Next
LL	Line length of text	6 inches (6i)	.nr LL 7.5i	paragraph
LT	Length of titles	LL	.nr LT 5i	page
FL	Line length of footnotes	5.5i	.nr FL LL	.FS request
FI	Footnote indent	5 ens (5n)	.nr FI 2n	.FS request
PS	Point size	10	.nr PS 11	paragraph
VS	Vertical spacing	12	.nr VS 13	paragraph
CW	Column width	LL * 7/15	.nr CW 3i	.2C or .MC request
GW	Intercolumn spacing	LL * 1/15	.nr GW .5i	.2C or .MC request
HM	Header margin	1i	.nr HM .75i	page
FM	Footer margin	1i	.nr FM .75i	page
PI	Paragraph indent	5 ens (5n)	.nr PI 2n	paragraph
PD	Paragraph depth	.3 vertical space (.3v)	.nr PD 0	paragraph
QI	Left and right indent for quote paragraph (.QP)	5n	.nr QI 8n	.QP request
DD	Vertical distance around displays	.5v	.nr DD 1v	.DS request
PO	Page offset	1i	.nr PO 0.5i	page
LH	Left page header	null	.ds LH Sun	page
CH	Center page header	null	.ds CH Confidential	page
RH	Right page header	null	.ds RH Software	page
LF	Left page footer	null	.ds LF Do Not Copy	page
CF	Center page footer	page number register (-\nPN-)	.ds CF Draft	page
RF	Right page footer	null	.ds RF %	page
%	Page number	1	.nr % 3	page

2

Formatting Documents with the -ms Macros

2

Formatting Documents with the -ms Macros

This chapter describes the -ms macro package for preparing documents with nroff and troff on the Sun system.[1] The -ms Request Summary at the end of this chapter provides a quick reference for all the -ms macros and for useful displaying and printing commands. If you are acquainted with -ms, there is a quick reference for the *new* requests and string definitions as well. The differences between the new and the old -ms macro packages are described in the section entitled "Changes in the New -ms Macro Package." The section "Displaying and Printing Documents with -ms" describes how you can produce documents on either your workstation, printer, or phototypesetter without changing the text and formatting request input.

2.1. Changes in the New -ms Macro Package

The old -ms macro package has been revised, and the new macro package assumes the name -ms. There are some extensions to previous -ms macros and a number of new macros, but all the previously documented -ms macros still work exactly as they did before, and have the same names as before. The new -ms macro package includes several bug fixes, including a problem with the single-column .1C macro, minor difficulties with boxed text, a break induced by .EQ before initialization, the failure to set tab stops in displays, and several bothersome errors in the refer bibliographic citation macros. Macros used only at Bell Laboratories have been removed from the new version. We list them at the end of this chapter in the

2.2. Displaying and Printing Documents with -ms

After you have prepared your document with text and -ms formatting requests and stored it in a file, you can display it on your workstation screen or print it with nroff or troff with the -ms option to use the -ms macro package. A good way to start is to pipe your file through more for viewing:

```
hostname% nroff -ms filename ...  | more
```

If you forget the -ms option, you get continuous, justified, unpaginated output in which -ms requests are ignored. You can format more than one file on the

[1] The material in this chapter is derived from *A Revised Version of* -ms, B. Tuthill, University of California, Berkeley; *Typing Documents on the UNIX System: Using the* -ms *Macros with* troff *and* nroff, M.E. Lesk, Bell Laboratories, Murray Hill, New Jersey; and *Document Formatting on UNIX: Using the* -ms *Macros*, Joel Kies, University of California, Berkeley.

command line at a time, in which case `nroff` simply processes all of them in the order they appear, as if they were one file. There are other *options* to use with `nroff` and `troff`; see the *SunOS Reference Manual* for details.

You can get preview and final output of various sorts with the following commands. To send `nroff` output to the line printer, type:

```
hostname% nroff -ms filename | lpr -Pprinter
```

To produce a file with tables, use:

```
hostname% tbl filename | nroff -ms | lpr -printer
```

To produce a file with equations, type:

```
hostname% neqn filename | nroff -ms | lpr -printer
```

To produce a file with tables and equations, use the following order:

```
hostname% tbl filename | neqn | nroff -ms | lpr -printer
```

To print your document with `troff`, use:

```
hostname% troff -ms filename | lpr -t -printer
```

See `lpr`(1) in the *SunOS Reference Manual* for details on printing.

2.3. What Can Macros Do?

Macros can help you produce paragraphs, lists, sections (optionally with automatic numbering), page titles, footnotes, equations, tables, two-column format, a table of contents, endnotes, running heads and feet, and cover pages for papers. As with other formatting utilities such as `nroff` and `troff`, you prepare text interspersed with formatting requests. However, the macro package, which itself is written in `troff` commands, provides higher-level commands than those provided with the basic `troff` program. In other words, you can do a lot more with just one macro than with one `troff` request.

2.4. Formatting Requests

An `-ms` request usually consists of one or two upper-case characters, and usually in the form *.XX*.

The easiest way to produce simple formatted text is to put a `.LP` request on a line by itself at the beginning of the document. Add your text, on the following lines, leaving just a blank line to separate paragraphs. The `.LP` request produces a left-blocked paragraph, as we used throughout this chapter. Your output will have paragraphs and be paginated with right and left-justified margins.

When you use a macro package, you type in text as you normally do and intersperse it with formatting *requests*. For example, instead of spacing in with the space bar or typing a tab to indent paragraphs, put a `.PP` request on a line by

itself before each paragraph. When formatted, this indents the first line of the following paragraph.

Note: You cannot just begin a document with a line of text. You must include an -ms request before any text input. When in doubt, use .LP to properly *initialize* the file, although any of the requests .PP, .LP, .TL, .SH, .NH is good enough. See the section "Cover Sheets and Title Pages" later in this chapter for the correct arrangement of requests at the start of a document.

Paragraphs

You can produce several different kinds of paragraphs with the -ms macro package: standard, left-block, indented, labeled, and quoted.

Standard Paragraph — .PP

To get an ordinary paragraph, use the .PP request, followed on subsequent lines by the text of the paragraph. For example, you type:

```
.PP
Tom appeared on the sidewalk with a bucket of whitewash and a long-handled
brush.
He surveyed the fence, and all gladness left him and a deep melancholy
settled down upon his spirit.
Thirty yards of board fence nine feet high.
Life to him seemed hollow, and
existence but a burden.
```

to produce:

Tom appeared on the sidewalk with a bucket of whitewash and a long-handled brush. He surveyed the fence, and all gladness left him and a deep melancholy settled down upon his spirit. Thirty yards of board fence nine feet high. Life to him seemed hollow, and existence but a burden.

Left-Block Paragraph — .LP

You can also produce a left-block paragraph, like those in this manual, with .LP. The first line is not indented as it is with the .PP request. For example, you type:

```
.LP
Tom appeared ...
```

to produce:

Tom appeared on the sidewalk with a bucket of whitewash and a long-handled brush. He surveyed the fence, and all gladness left him and a deep melancholy settled down upon his spirit. Thirty yards of board fence nine feet high. Life to him seemed hollow, and existence but a burden.

There are default values for the vertical spacing before paragraphs and for the width of the indentation. To change the paragraph spacing, see the section "Modifying Default Features."

Indented Paragraph — .IP

Another kind of paragraph is the indented paragraph, produced by the .IP request. These paragraphs can have hanging numbers or labels. For example:

```
.IP [1]
Text for first paragraph, typed
normally for as long as you would
like on as many lines as needed.
.IP [2]
Text for second paragraph, ...
.LP
```

produces

[1] Text for first paragraph, typed normally for as long as you would like on as many lines as needed.

[2] Text for second paragraph, ...

A series of indented paragraphs must be followed by an ordinary paragraph beginning with .PP or .LP, depending on whether you wish indenting or not. Here we used the .LP request.

More sophisticated uses of .IP are also possible. If the label is omitted, for example, you get a plain block indent:

```
Tom appeared on the sidewalk with a bucket of whitewash and a long-handled
brush.
.IP
He surveyed the fence, and all gladness left him and a deep melancholy
settled down upon his spirit.
Thirty yards of board fence nine feet high.
Life to him seemed hollow, and
existence but a burden.
.LP
```

which produces

Tom appeared on the sidewalk with a bucket of whitewash and a long-handled brush.

> He surveyed the fence, and all gladness left him and a deep melancholy settled down upon his spirit. Thirty yards of board fence nine feet high. Life to him seemed hollow, and existence but a burden.

If a non-standard amount of indenting is required, specify it after the label in character positions. It remains in effect until the next .PP or .LP. Thus, the general form of the .IP request contains two additional fields: the label and the indenting length. For example,

```
.IP "Example one:" 15
Notice the longer label, requiring larger
indenting for these paragraphs.
.IP "Example two:"
And so forth.
.LP
```

produces this:

Example one: Notice the longer label, requiring larger indenting for these paragraphs.

Example two: And so forth.

Notice that you must enclose the label in double quote marks because it contains a space; otherwise, the space signifies the end of the argument. The indentation request above is in the number of *ens*, a unit of dimension used in typesetting. An *en* is approximately the width of a lowercase 'n' in the particular point size you are using.

The `.IP` macro adjusts properly by causing a break to the next line if you type in a label longer than the space you allowed for. For example, if you have a very long label and have allowed 10 en-spaces for it, your input looks like:

```
.IP "A very, very, long and verbose label" 10
And now here's the text that you want.
And now here's the text that you want.
And now here's the text that you want.
And now here's the text that you want.
And now here's the text that you want.
```

And your output is adjusted accordingly with a break between the label and the text body:

A very, very, long and verbose label
And now here's the text that you want. And now here's the text that you want. And now here's the text that you want. And now here's the text that you want. And now here's the text that you want.

Nested Indentation — `.RS` and `.RE`

It is also possible to produce multiple (or *relative*) nested indents; the `.RS` request indicates that the next `.IP` starts its indentation from the current indentation level. Each `.RE` undoes one level of indenting, so you should balance `.RS` and `.RE` requests. Think of the `.RS` request as 'move right' and the `.RE` request as 'move left'. As an example:

```
.IP I.
South Bay Area Restaurants
.RS
.IP A.
Palo Alto
.RS
.IP 1.
La Terrasse
.RE
.IP B.
Mountain View
.RS
.IP 1.
Grand China
.RE
.IP C.
Menlo Park
.RS
.IP 1.
Late for the Train
.IP 2.
Flea Street Cafe
.RE
.RE
.LP
```

results in:

I. South Bay Area Restaurants

   A. Palo Alto

      1. La Terrasse

   B. Mountain View

      1. Grand China

   C. Menlo Park

      1. Late for the Train

      2. Flea Street Cafe

Note the two `.RE` requests in a row at the end of the list. Remember that you need one *end* for each *start*.

Quoted Paragraph — `.QP`

All of the variations on `.LP` leave the right margin untouched. Sometimes, you need a a paragraph indented on both right and left sides. To set off a quotation as such, use:

```
.QP
Precede each paragraph that you want offset as a quotation
with a .QP. This produces a paragraph like this.
Notice that the right edge is also indented from the right margin.
```

to produce

> Precede each paragraph that you want offset as a quotation with a .QP. This produces a paragraph like this. Notice that the right edge is also indented from the right margin.

Section Headings — .SH and .NH

There are two varieties of section headings, unnumbered with .SH and numbered with .NH. In either case, type the text of the section heading on one or more lines following the request. End the section heading by typing a subsequent paragraph request or another section heading request. When printed, one line of vertical space precedes the heading, which begins at the left margin. nroff offsets the heading with blank lines, while troff sets it in **boldface** type. .NH section headings are numbered automatically. The macro takes an argument number representing the *level-number* of the heading, up to 5. A third-level section number is one like '1.2.1'. The macro adds one to the section number at the requested level, as shown in the following example:

```
.NH
Bay Area Recreation
.NH 2
Beaches
.NH 3
San Gregorio
.NH 3
Half Moon Bay
.NH 2
Parks
.NH 3
Wunderlich
.NH 3
Los Trancos
.NH 2
Amusement Parks
.NH 3
Marine World/Africa USA
```

generates:

2. Bay Area Recreation

2.1 Beaches

2.1.1 San Gregorio

2.1.2 Half Moon Bay

2.2 Parks

2.2.1 Wunderlich

2.2.2 Los Trancos

2.3 Amusement Parks

2.3.1 Marine World/Africa USA

`.NH` without a level-number means the same thing as `.NH 1`, and `.NH 0` cancels the numbering sequence in effect and produces a section heading numbered 1.

Cover Sheets and Title Pages — `.TL` and `.AU`

`-ms` provides a group of macros to format items that typically appear on the cover sheet or title page of a formally laid-out paper. You can use them selectively, but if you use several, you must put them in the order shown below, normally at or near the beginning of the input file.

The first line of a document signals the general format of the first page. In particular, if it is `.RP` (released paper), a cover sheet with title and abstract is prepared. The default format is useful for scanning drafts.

Sample input is:

```
.RP       (Optional; use for released paper format)
.TL
Title of document (one or more lines)
.AU
Author(s)       (may also be several lines)
.AI
Author's institution(s)
.AB
Abstract; to be placed on the cover sheet of a paper.
Line length is 5/6 of normal; use .ll here to change.
.AE       (abstract end)
text ...        (begins with .PP)
```

(See *Order of Requests in Input* for a quick example of this scheme.)

If the `.RP` request precedes `.TL`, the title, author, and abstract material are printed separately on a cover sheet. The title and author information (not the abstract) is then repeated automatically on page one (the title page) of the paper, without your having to type it again. If you do not include an `.RP` request, all of this material appears on page one, followed on the same page by the main text of the paper.

To omit some of the standard headings (such as no abstract, or no author's institution), just omit the corresponding fields and command lines. To suppress the word ABSTRACT type `.AB no` rather than `.AB`. You can intersperse several `.AU` and `.AI` lines to format for multiple authors.

These macros are optional; you may begin a paper simply with a section heading or paragraph request. When you do precede the main text with cover sheet and title page material, include a paragraph or section heading between the last title page request and the beginning of the main text. Don't forget that some `-ms` request must precede any input text.

Running Heads and Feet — LH, CH, RH

The `-ms` macros, by default, print a page heading containing a page number (if greater than 1). You can make minor adjustments to the page headings and footings by redefining the strings LH, CH, and RH which are the left, center and right portions of the page headings, respectively; and the strings LF, CF, and RF, which are the left, center and right portions of the page footer. For `nroff` output, there are two default values: CH is the current page number surrounded on either side by hyphens, and CF contains the current date as supplied by the computer. For `troff` CH also contains the page number, but CF is empty. The other four registers are empty by default for both `nroff` and `troff`. You can use the `.ds` request to assign a value to a string register. For example:

```
.ds RF Draft Only \(em Do Not Distribute
```

This prints the character string

at the bottom right of every page. You do not need to enclose the string in double quote marks. To remove the contents of a string register, simply redefine it as empty. For instance, to clear string register CH, and make the center header blank on the following pages, use the request:

```
.ds CH
```

To put the page number in the right header, use:

```
.ds RH %
```

In a string definition, '%' is a special symbol referring to `nroff`'s automatic page counter. If you want hyphens on either side of the page number, place them on either side of the '%' in the command, that is:

```
.ds RH -%-
```

Remember that putting the page number in the right header as shown above does not remove it from the default CH; you still have to clear out CH.

If you want requests that set the values of string and number registers to take effect on the first page of output, put them at or near the beginning of the input file, before the initializing macro, which in turn must precede the first line of text. Among other functions, the initializing macro causes a 'pseudo page break' onto page one of the paper, including the top-of-page processing for that page. Be sure to put requests that change the value of the PO (page offset), HM (top or head margin), and FM (bottom or foot margin) number registers and the page header string registers before the transition onto the page where they are to take effect.

For more complex formats, you can redefine the macros PT (page top) and BT (page bottom), which are invoked respectively at the top and bottom of each page. The margins (taken from registers HM and FM for the top and bottom margin respectively) are normally 1 inch; the page header/footer are in the middle of that space. If you redefine these macros, be careful not to change parameters such as point size or font without resetting them to default values.

Custom Headers and Footers — `.OH`, `.EH`, `.OF`, and `.EF`

You can also produce custom headers and footers that are different on even and odd pages. The `.OH` and `.EH` macros define odd and even headers, while `.OF` and `.EF` define odd and even footers. Arguments to these four macros are specified as with the `nroff .tl`, that is, there are three fields (left, center and right), each separated by a single apostrophe. For example, to get odd-page headers with the chapter name followed by the page number and the reverse on even pages, use:

```
.OH  ' For Whom the Bell Tolls' ' Page  %'
.EH  ' Page  %' ' For Whom the Bell Tolls'
```

Note that it is an error to have an apostrophe in the header text; if you need an apostrophe, use a backslash and apostrophe (\') or a delimiter other than apostrophe around the left, center, and right portions of the title. You can use any character as a delimiter, provided it doesn't appear elsewhere in the argument to `.OH`, `.EH`, `.OF`, or `.EF`.

You can use the `.P1` (dot-P-one) macro to print the header on page 1. If you want roman numeral page numbering, use an `.af PN i` request.

Multi-Column Formats — `.2C` and `.MC`

If you place the request `.2C` in your document, the document will be printed in double column format beginning at that point. This is often desirable on the typesetter. Each column will have a width 7/15 that of the text line length in single-column format, and a gutter (the space between the columns) of 1/15 of the full line length. Remember that when you use the two-column `.2C` request, either pipe the `nroff` output through `col` or make the first line of the input `.pi /usr/bin/col`.

The `.2C` request is actually a special case of the `.MC` request that produces formats of more than two spaces:

`.MC` [*column width* [*gutter width*]]

This formats output in as many columns of *column width* as will fit across the page with a gap of *gutter width*. You can specify the column width in any unit of scale, but if you do not specify a unit, the setting defaults to ens. `.MC` without any column width is the same thing as `.2C`. For example:

```
.MC
Tom appeared on the sidewalk with a bucket of whitewash and a long-handled
brush.
He surveyed the fence, and all gladness left him and a deep melancholy
settled down upon his spirit.
```

To return to single-column output, use `.1C`. Switching from double to single-column always causes a skip to a new page.

Footnotes — .FS and .FE

Material placed between lines with the commands .FS (footnote start) and .FE (footnote end) is collected, remembered, and placed at the bottom of the current page.* The formatting of the footnote is:

```
...at the bottom of the current page.*
.FS
* Like this.
.FE
```

By default, footnotes are 11/12th the length of normal text, but you can modify this by changing the FL register (see the "Modifying Default Features" section). When typeset, footnotes appear in smaller size type.

Because the macros only save a passage of text for printing at the bottom of the page, you have to mark the footnote reference in some way, both in the text preceding the footnote and again as part of the footnote text. We use a simple asterisk, but you can use anything you want.

You can also produce automatically-numbered footnotes. Footnote numbers are printed by a predefined string (**), which you invoke separately from .FS and .FE. Each time this string is used, it increases the footnote number by one, whether or not you use .FS and .FE in your text. Footnote numbers are superscripted on the phototypesetter and on daisy-wheel terminals, but on low-resolution devices (such as the line printer and a treminal), they are bracketed. If you use ** to indicate numbered footnotes, the .FS macro automatically includes the footnote number at the bottom of the page. This footnote, for example, was produced as follows:[2]

```
This footnote, for example, was produced as follows:\**
.FS
If you never use the ...
.FE
```

If you are using ** to number footnotes, but want a footnote of the same style marked with an asterisk or dagger, give that mark as the first argument to .FS:†

```
give that mark as the first argument to .FS:\(dg
.FS \(dg
In the footnote, the dagger ...
.FE
```

Footnote numbering is temporarily suspended, because the ** string is not used. Instead of a dagger, you could use an asterisk * or double dagger ‡, represented as \(dd.

---

* Like this.

[2] If you never use the ** string, no footnote numbers will appear anywhere in the text, including down here. The output footnotes will look exactly like footnotes produced with **-mos**, the old -ms macro package.

† In the footnote, the dagger will appear where the footnote number would otherwise appear, as shown here.

Endnotes

If you want to produce endnotes rather than footnotes, put the references in a file of their own. This is similar to what you would do if you were typing the paper on a conventional typewriter. Note that you can use automatic footnote numbering without actually having the `.FS` and `.FE` pairs in your text. If you place footnotes in a separate file, you can use `.IP` macros with `\**` as a hanging tag; this gives you numbers at the left-hand margin. With some styles of endnotes, you would want to use `.PP` rather than `.IP` macros, and specify `\**` before the reference begins.

Displays and Tables — `.DS` and `.DE`

To prepare displays of lines, such as tables, in which the lines should not be rearranged or broken between pages, enclose them in the requests `.DS` and `.DE`:

```
.DS
lines, like the
examples here, are placed
between .DS and .DE macros
.DE
```

which produces:

lines, like the
examples here, are placed
between .DS and .DE macros

By default, lines between `.DS` and `.DE` are indented from the left margin.

If you don't want the indentation, use `.DS L` to begin and `.DE` to produce a left-justified display:

to get
something like
this

You can also center lines with the `.DS C` and `.DE` requests:

This is an
example
of a centered display.

Note that each line is centered individually.

A plain `.DS` is equivalent to `.DS I`, which indents and left-adjusts. An extra argument to the `.DS I` or `.DS` request is taken as an amount to indent. For example, `.DS I 3` or `.DS 3` begins a display to be indented 3 ens from the margin.

There is a variant `.DS B` that makes the display into a left-adjusted block of text, and then centers that entire block.

Normally a display is kept together on one page. If you wish to have a long display which may be split across page boundaries, use `.CD`, `.LD`, and `.BD` in place of the requests `.DS C`, `.DS L`, and `.DS B` respectively. Use `.ID` for either a plain `.DS` or `.DS I`. You can also specify the amount of indentation with the `.ID` macro. Use the following table as a quick reference:

Table 2-1 *Display Macros*

Macro with Keep	Macro without Keep
.DS I	.ID
.DS L	.LD
.DS C	.CD
.DS B	.BD
.DS	.ID

Note: It is tempting to assume that `.DS R` will right-adjust lines, but it doesn't.

Keeping Text Together — `.KS`, `.KE` and `.KF`

If you wish to keep a table or other block of lines together on a page, there are 'keep - release' requests. If a block of lines preceded by `.KS` and followed by `.KE` does not fit on the remainder of the current page, it will begin on a new page. There is also a 'keep floating' request. If the block to be kept together is preceded by `.KF` instead of `.KS` and does not fit on the current page, it will be moved down through the text to the top of the next page. `nroff` fills in the current page with the ordinary text that follows the keep in the input file to avoid leaving blank space at the bottom of the page preceding the keep. Thus, no large blank space will be introduced in the document.

In multi-column output, the keep macros attempt to place all the kept material in the same column. If the material enclosed in a keep requires more than one page, or more than a column in multi-column format, it will start on a new page or column and simply run over onto the following page or column.

Boxing Words or Lines — `.BX` and `.B1` and `.B2`

To draw rectangular boxes around words, use the request

```
.BX word
```

to print word as shown. You can box longer pieces of text by enclosing them with `.B1` and `.B2`:

```
.B1
Tom appeared on the sidewalk with a bucket of whitewash
and a long-handled brush.
He surveyed the fence, and all gladness left him and a deep melancholy
settled down upon his spirit.
Thirty yards of board fence nine feet high.
Life to him seemed hollow, and
existence but a burden.
.B2
```

This produces:

Tom appeared on the sidewalk with a bucket of whitewash and a long-handled brush. He surveyed the fence, and all gladness left him and a deep melancholy settled down upon his spirit. Thirty yards of board fence nine feet high. Life to him seemed hollow, and existence but a burden.

Changing Fonts — .I, .B, .R and .UL

To get italics on the typesetter or reverse display on the workstation, say:

```
.I
as much text as you want
can be typed here
.R
```

as was done for *these three words*. The .R request restores the normal (usually Roman) font. If only one word is to be italicized, you can put it on the line with the .I request:

```
.I word
```

and in this case you do not need to use an .R to restore the previous font.

You can print **boldface** font by

```
.B
Text to be set in boldface
goes here
.R
```

As with .I, you can place a single word in boldface font by putting it on the same line as the .B request. Also, when .I or .B is used with a word as an argument, it can take as a second argument any trailing punctuation to be printed immediately after the word but set in normal typeface. For example:

```
.B word )
```

prints

word)

that is, the word in boldface and the closing parenthesis in normal Roman directly adjacent to the word.

If you want actual underlining as opposed to italicizing on the typesetter, use the request

```
.UL word
```

to underline a word. There is no way to underline multiple words on the typesetter.

Changing the Type Size — .LG, .SM and .NL

You can specify a few size changes in troff output with the requests .LG (make larger), .SM (make smaller), and .NL (return to normal size). The size change is two points (see the "Dimensions" section for a discussion of point size); you can repeat the requests for increased effect (here one .NL canceled two .SM requests). These requests are primarily useful for temporary size changes

for a small number of words. They do not affect vertical spacing of lines of text. See the section on "Modifying Default Features" for other techniques for changing the type size and vertical spacing of longer passages.

Dates — .DA and .ND

When you use -ms, nroff prints the date at the bottom of each page, but troff does not. Both nroff and troff print it on the cover sheet if you have requested one with .RP. To make troff print the date as the center page footer, say .DA (date). To suppress the date, say .ND (no date). To lie about the date, type .DA July 4, 1776, which puts the specified date at the bottom of each page. The request:

```
.ND September 16, 1959
```

in .RP format places the specified date on the cover sheet and nowhere else. Place either .ND or .DA before the .RP Notice this is one instance that you do not need to put double quote marks around the arguments.

Thesis Format Mode — .TM

To format a paper as a thesis, use the .TM macro (thesis mode). It is much like the .th macro in the **-me** macro package. It puts page numbers in the upper right-hand corner, numbers the first page, suppresses the date, and doublespaces everything except quotes, displays, and keeps. Use it at the top of each file making up your thesis. Calling .TM defines the .CT macro for chapter titles, which skips to a new page and moves the page number to the center footer. You can use the .P1 (P one) macro even without thesis mode to print the header on page one, which is suppressed except in thesis mode. If you want roman numeral page numbering, use an .af PN i request.

Bibliography — .XP

To format bibliography entries, use the .XP macro, which stands for *exdented paragraph.* It exdents the first line of the paragraph by \n(PI units, usually 5n, the same as the indent for the first line of a .PP. An example of exdented paragraphs is:

```
.XP
Lumley, Lyle S., \fISex in Crustaceans: Shell Fish Habits,\fP\^
Harbinger Press, Tampa Bay and San Diego, October 1979.
243 pages.
The pioneering work in this field.
.XP
Leffadinger, Harry A., "Mollusk Mating Season: 52 Weeks, or All Year?"
in \fIActa Biologica,\fP\^ vol. 42, no. 1J, November 1980.
A provocative thesis, but the conclusions are wrong.
```

which produces:

Lumley, Lyle S., *Sex in Crustaceans: Shell Fish Habits,* Harbinger Press, Tampa Bay and San Diego, October 1979. 243 pages. The pioneering work in this field.

Leffadinger, Harry A., "Mollusk Mating Season: 52 Weeks, or All Year?" in *Acta Biologica,* vol. 42, no. 11, November 1980. A provocative thesis, but the conclusions are wrong.

You do have to italicize the book and journal titles and quote the title of the journal article. You can change the indentation and exdentation by setting the value of number register PI.

Table of Contents — .XS, .XE, .XA, .PX

There are four macros that produce a table of contents. Enclose table of contents entries in .XS and .XE pairs, with optional .XA macros for additional entries. Arguments to .XS and .XA specify the page number, to be printed at the right. A final .PX macro prints out the table of contents. A sample of typical input and output text is:

```
.XS  ii
Introduction
.XA  1
Chapter 1: Review of the Literature
.XA  23
Chapter 2: Experimental Evidence
.XE
.PX
```

Table of Contents

You can also use the .XS and .XE pairs in the text, after a section header for instance, in which case page numbers are supplied automatically. However, most documents that require a table of contents are too long to produce in one run, which is necessary if this method is to work. It is recommended that you make the table of contents after finishing your document. To print out the table of contents, use the .PX macro or nothing will happen.

Defining Quotation Marks

To produce quotation marks and dashes that format correctly with both nroff and troff, there are some string definitions for each of the formatting programs. The *— string yields two hyphens in nroff, and produces an em-dash — like this one in troff. The *Q and *U strings produce " and " in troff, but " in nroff.

Accent Marks

To simplify typing certain foreign words, the -ms macro package defines strings representing common accent marks. There are a large number of optional foreign accent marks defined by the -ms macros. All the accent marks available in **-mos** are present, and they all work just as they always did.

For the old accent marks, type the string *before* the letter over which the mark is to appear. For example, to print 't'el'ephone with the old macros, you type:

```
t\*'el\*'ephone
```

Unlike the old accent marks, the new accent strings should be placed *after* the letter being accented. Place `.AM` (accent mark) at the beginning of your document, and type the accent strings *after* the letter being accented. A list of both sets of diacritical marks and examples of what they look like follows. *Note:* Do not use the `tbl` macros `.TS` and `.TE` with any of the accent marks as the marks do not line up correctly.

Table 2-2 *Old Accent Marks*

Accent Name	Input	Output
acute	*'e	é
grave	*`e	è
umlaut	*:u	ü
circumflex	*^e	ê
tilde	*~a	ã
hacek	*Cr	ř
cedilla	*,c	c

Table 2-3 *Accent Marks*

Accent Name	Input	Output
acute	e*'	é
grave	e*`	è
circumflex	o*^	ô
cedilla	c*,	ç
tilde	n*~	ñ
question	*?	¿
exclamation	*!	¡
umlaut	u*:	ü
digraphes	*8	ß
hacek	c*v	č
macron	a*_	ā
o-slash	o*/	ø
yogh	kni*3t	kniȝt
angstrom	a*o	å
Thorn	*(Th	Þ
thorn	*(th	þ
Eth	*(D-	Ð
eth	*(d-	ð
hooked o	*q	ǫ
ae ligature	*(ae	æ
AE ligature	*(Ae	Æ
oe ligature	*(oe	œ
OE ligature	*(Oe	Œ

If you want to use these new diacritical marks, don't forget the `.AM` at the top of your file. Without it, some of these marks will not print at all, and others will be placed on the wrong letter.

2.5. Modifying Default Features

The -ms macro package supplies a standard page layout style. The text line has a default length of six inches; the indentation of the first line of a paragraph is five ens; the page number is printed at the top center of every page after page one; and so on for standard papers. You can alter many of these default features by changing the values that control them.

The computer memory locations where these values are stored are called *number registers* and *string registers*. Number and string registers have names like those of requests, one or two characters long. For instance, the value of the line length is stored in a number register named LL. Unless you give a request to change the value stored in register LL, it will contain the standard or default value assigned to it by -ms. The "Summary of -ms Number Registers" table lists the number registers you can change along with their default values.

Dimensions

To change a dimension like the line length from its default value, reset the associated number register with the troff request .nr (number register):

```
.nr LL 5i
```

The first argument, LL, is the name of a number register, and the second, 5i is the value being assigned to it. In the case above, the line length is adjusted from the default six inches to five inches. As another example, consider:

```
.nr PS 9
```

which makes the default point size 9 point.

The value may be expressed as an integer or may contain a decimal fraction. When setting the value of a number register, it is almost always necessary to include a unit of scale immediately after the value. In the example above, the 'i' as the unit of scale lets troff know you mean five inches and not five of some other unit of distance. But the point size (PS) and vertical spacing (VS) registers are exceptions to this rule; ordinarily they should be assigned a value as a number of points *without indicating the unit of scale.* For example, to set the vertical spacing to 24 points, or one-third of an inch (double-spacing), use the request:

```
.nr VS 24
```

In the unusual case where you want to set the vertical spacing to more than half an inch (more than 36 points), include a unit of scale in setting the VS register. The "Units of Measurement in nroff and troff" table explains the units of measurement.

Table 2-4 *Units of Measurement in* `nroff` *and* `troff`

Unit	*Abbr*	`nroff`	`troff`
point	p	1/72 inch	1/72 inch
pica	p	1/6 inch	1/6 inch
em	m	width of one character	distance equal to number of points in the current typesize
en	n	width of one character	half an em
vertical space	v	amount of space in which each line of text is set, measured baseline to baseline	same
inch	i	inch	inch
centimeter	c	centimeter	centimeter
machine unit	u	1/240 inch	1/432 inch

The units *point*, *pica*, *em*, and *en* are units of measurement used by tradition in typesetting. The *vertical space* unit also corresponds to the typesetting term *leading*, which refers to the distance from the baseline of one line of type to the baseline of the next. Em and en are particularly interesting in that they are proportional to the type size currently in use (normally expressed as a number of points). An em is the distance equal to the number of points in the type size (roughly the width of the letter 'm' in that point size), while an en is half that (about the width of the letter 'n'). These units are convenient for specifying dimensions such as indentation. In `troff`, em and en have their traditional meanings, that is one em of distance is equal to two ens. For `nroff`, on the other hand, em and en both mean the same quantity of distance, the width of one typewritten character.

The *machine unit* is a special unit of dimension used by `nroff` and `troff` internally. This is the unit to which the programs convert almost all dimensions when storing them in memory, and is included here primarily for completeness. In using the features of `-ms`, it is sufficient to know that such a unit of measure exists.

Note that a change to a number register such as LL does not immediately change the related dimension at that point in the output. Instead, in the case of the line length for example, the change takes place at the beginning of the next paragraph, where `-ms` resets various dimensions to the current values of the related number registers.

If you need the effect immediately, use the normal `troff` command in addition to changing the number register. For example, to control the vertical spacing immediately, use:

```
.vs
```

This takes effect at the place where it occurs in your input file. Since it does not

change the VS register, however, its effect lasts only until the beginning of the next paragraph. As a general rule, to make a permanent change, or one that will last for several paragraphs until you want to change it again, alter the value of the `-ms` register. If the change must happen immediately, somewhere other than the point shown in the table, use the `troff` request. If you want the change to be both immediate and lasting, do both.

Table 2-5 *Summary of* `-ms` *Number Registers*

Register	Controls	Takes Effect	Default
PS	point size	next para.	10
VS	line spacing	next para.	12 pts
LL	line length	next para.	6″
LT	title length	next para.	6″
PD	para. spacing	next para.	0.3 VS
PI	para. indent	next para.	5 ens
FL	footnote length	next `.FS`	11/12 LL
CW	column width	next `.2C`	7/15 LL
GW	intercolumn gap	next `.2C`	1/15 LL
PO	page offset	next page	26/27″
HM	top margin	next page	1″
FM	bottom margin	next page	1″

You may also alter the strings and which are the left, center, and right headings respectively; and similarly and which are strings in the page footer. Use the `troff .ds` (define string) request to alter the string registers, as you use the `.nr` request for number registers. The page number on *output* is taken from register to permit changing its output style. For more complicated headers and footers, you can redefine the macros and as explained earlier. See the "Register Names" section for a full list.

2.6. Using `nroff` and `troff` Requests

You can use a small subset of the `troff` requests to supplement the `-ms` macro package.

Use `.nr` and `.ds` requests to manipulate the `-ms` number and string registers as described in the "Modifying Default Features" section. You can also freely use the other following requests in a file for processing with the `-ms` macro package. They all work with both typesetter and workstation or terminal output.

`.ad b` Adjust both margins. This is the default adjust mode.

`.bp` Begin new page.

`.br` 'Break' line; start a new output line whether or not the current one has been completely filled with text.

`.ce` *n* Center the following *n* input text lines individually in the output. If *n* is omitted, only the next (one) line of text is centered.

`.ds` *XX* Define string register named *XX*.

`.na` Turn off adjusting of right margins to produce ragged right.

`.nr` *XX* Define number register named XX.

`.sp` *n* Insert *n* blank lines. If *n* is omitted, one blank line is produced (the current value of the unit *v*). You can attach a unit of dimension to *n* to specify the quantity in units other than a number of blank lines.

Note: The macro package executes sequences of `troff` requests on its own, in a manner invisible to you. By inserting your own `troff` requests, you run the risk of introducing errors. The most likely result is simply for your `troff` requests to be ignored, but in some cases the results can include fatal `troff` errors and garbled typesetter output.

As a simple example, if you try to produce a centered heading with the input:

```
.ce
.SH
Text of section heading
```

you will discover that the heading comes out left-adjusted; the `.SH` macro, appearing after the `.ce` request overrules it and forces left-adjusting. But consider the following sequence:

```
.sp
.ce
.B
Line of text
```

which successfully produces a centered, boldface heading preceded by one line of vertical space. There are lots of tricks like this, so be careful.

To learn more about `troff` see the chapter on "Formatting Documents with `nroff` and `troff`."

2.7. Using `-ms` with `eqn` to Typeset Mathematics

If you have to print Greek letters or mathematical equations, see the chapter "Typesetting Mathematics with `eqn`" for equation setting. To aid `eqn` users, `-ms` provides definitions of `.EQ` and `.EN` which normally center the equation and set it off slightly. An argument to `.EQ` is taken to be an equation number and placed in the right margin near the equation. In addition, there are three special arguments to `.EQ`: the letters `C`, `I`, and `L` indicate centered (default), indented, and left adjusted equations, respectively. If there is both a format argument and an equation number, give the format argument first, as in

```
.EQ L (1.3a)
```

for a left-adjusted equation numbered (1.3a).

2.8. Using -ms with tbl to Format Tables

Similar to the eqn macros are the macros .TS and .TE defined to separate tables from text with a little space (see the chapter "Formatting Tables with tbl"). A very long table with a heading may be broken across pages by beginning it with .TS H instead of .TS, and placing the line .TH in the table data after the heading. If the table has no heading repeated from page to page, just use the ordinary .TS and .TE macros.

2.9. Register Names

The -ms macro package uses the following register names internally. Independent use of these names in your own macros may produce incorrect output. Note that there are no lower case letters in any -ms internal name.

Number Registers Used in -ms

:	DW	GW	HM	IQ	LL	NA	OJ	PO	T.	TV
#T	EF	H1	HT	IR	LT	NC	PD	PQ	TB	VS
T.	FC	H2	IF	IT	MF	ND	PE	PS	TC	WF
1T	FL	H3	IK	KI	MM	NF	PF	PX	TD	YE
AV	FM	H4	IM	L1	MN	NS	PI	RO	TN	YY
CW	FP	H5	IP	LE	MO	OI	PN	ST	TQ	ZN

String Registers Used in -ms

'	A5	CB	DW	EZ	I	KF	MR	R1	RT	TL
`	AB	CC	DY	FA	I1	KQ	ND	R2	S0	TM
^	AE	CD	E1	FE	I2	KS	NH	R3	S1	TQ
~	AI	CF	E2	FJ	I3	LB	NL	R4	S2	TS
:	AU	CH	E3	FK	I4	LD	NP	R5	SG	TT
,	B	CM	E4	FN	I5	LG	OD	RC	SH	UL
1C	BG	CS	E5	FO	ID	LP	OK	RE	SM	WB
2C	BT	CT	EE	FQ	IE	ME	PP	RF	SN	WH
A1	C	D	EL	FS	IM	MF	PT	RH	SY	WT
A2	C1	DA	EM	FV	IP	MH	PY	RP	TA	XD
A3	C2	DE	EN	FY	IZ	MN	QF	RQ	TE	XF
A4	CA	DS	EQ	HO	KE	MO	R	RS	TH	XK

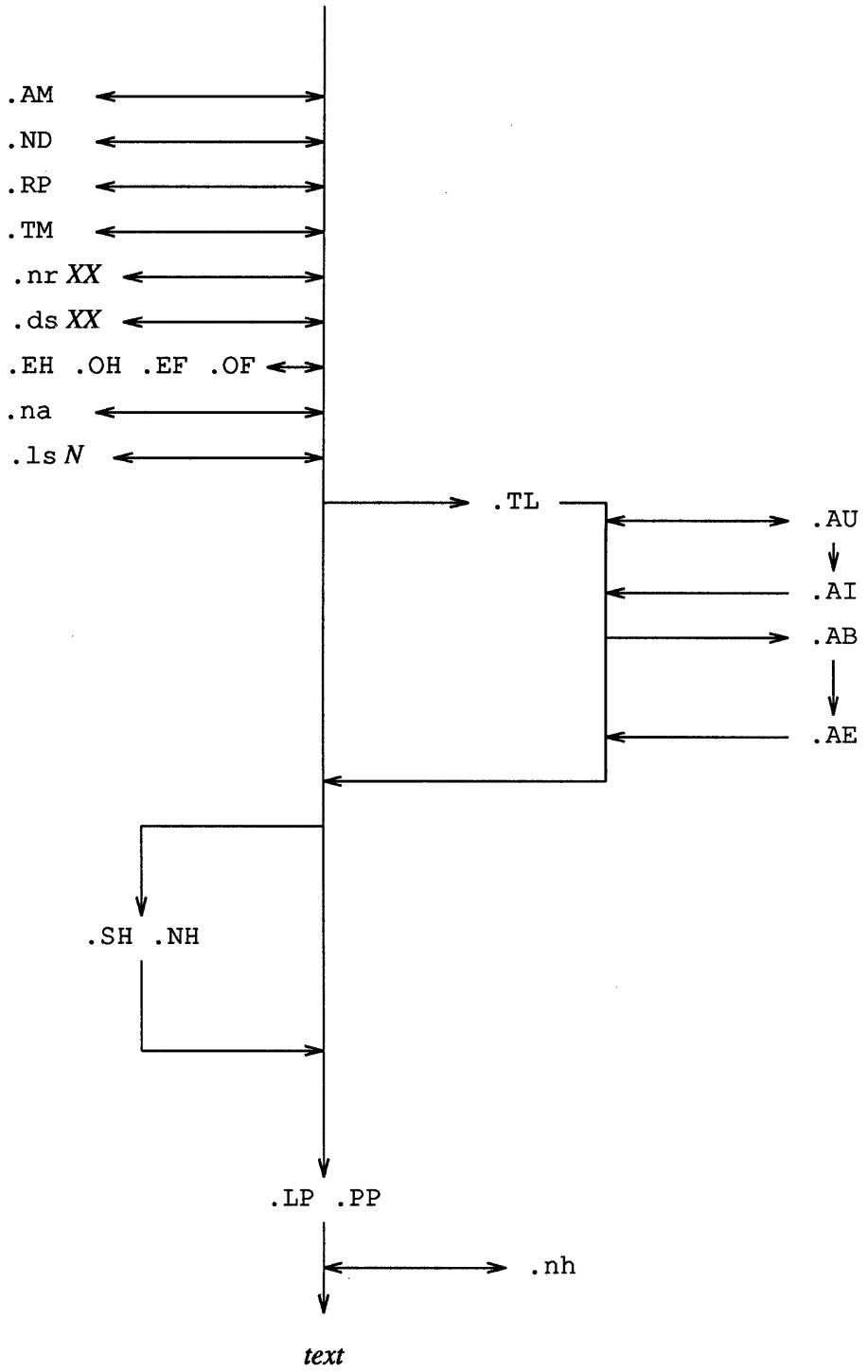
2.10. Order of Requests in Input

The following diagram provides a quick reference for how to order macro requests when using the -ms macro package to format documents. The central arrow indicates that the minimum formatting requests you need with -ms are the paragraph macros. These initialize certain quantities and are necessary to obtain predictable results when you use other macros.

The double-edged arrows indicate optional requests. The single-edged arrows indicate dependencies. For example, if you use a .AB request, you need a .AE request. If you use a .AU request, you don't need a .AI request, but if you use a .AI request, you have to use a .AU request first. The locations of the side arrows relative to the other requests indicate the relative locations of the requests in the document source.

For simpler documents, use just a `.LP` initializing request and just leave blank lines between paragraphs.

Figure 2-1 *Order of Requests in* `-ms` *Documents*



2.11. -ms Request Summary

This section includes tables of the old Bell Laboratories that have been removed from the new -ms package, of new -ms requests and string definitions, and of useful printing and displaying commands. It also includes a complete -ms request and string summary for easy reference.

Table 2-6 *Bell Laboratories Macros Deleted From* -ms

Macro Request	Explanation
.CS	Cover sheet
.EG	BTL Engineer's Notes
.HO	Bell Labs, Holmdel, N.J.
.IH	Bell Labs, Naperville, Ill.
.IM	BTL internal memo
.MF	BTL file memo
.MH	Bell Labs, Murray Hill, N.J.
.MR	BTL record memo
.ND	BTL date
.OK	BTL keywords for tech memo
.PY	Bell Labs, Piscataway, N.J.
.SG	Signatures for tech memo
.TM	BTL technical memo
.TR	BTL report format
.WH	Bell Labs, Whippany, N.J.

Table 2-7 *New* -ms *Requests*

Macro Request	Explanation
.AM	New accent mark definitions.
.CT	Chapter title in . TM format.
.EH	Define even three-part page header.
.EF	Define even three-part page footer.
.FE	End automatically numbered footnote.
.FS	Begin automatically numbered footnote.
.IP **	Endnotes with automatic numbering.
.IX	Index words.
.OF	Define odd three-part page footer.
.OH	Define odd three-part page header.
.P1	Put header on page one in . TM format.
.PX	Print table of contents.
.TM	Thesis mode format.
.XS	Start table of contents entry.
.XE	End table of contents entry.
.XA	Additional table of contents entry.
.PX	Prints table of contents.
.XP	Exdented paragraph.

Table 2-8 *New String Definitions*

Definition	In `nroff`	In `troff`
`\*-`	Two dashes --	Em dash —
`\*Q`	Open quote "	Open quote “
`\*U`	Close quote "	Close quote ”

Table 2-9 `-ms` *Macro Request Summary*

Macro Request	Initial Value	Cause Break?	Explanation
`.1C`	yes	yes	One column format on a new page.
`.2C`	no	yes	Two column format.
`.AB`	no	yes	Begin abstract.
`.AE`	–	yes	End abstract.
`.AI`	no	yes	Author's institution follows.
`.AM`	–	no	New accent mark definitions
`.AT`	no	yes	Print '...Attached' and turn off line filling.
`.AU`	no	yes	Author's name follows.
`.B` *x*	no	no	Print *x* in boldface; if no argument switch to boldface.
`.B1`	no	yes	Begin text to be enclosed in a box.
`.B2`	no	yes	End text to be boxed and print it.
`.BT`	date	no	Bottom title, automatically invoked at foot of page. May be redefined.
`.BX` *x*	no	no	Print *x* in a box.
`.CM`	-	no	Cut mark between pages (only if troff).
`.CT`	-	yes	Chapter title in thesis mode only. Page number moved to CF.
`.DA` *x*	date	no	'Date line' at bottom of page is *x* (only in `nroff`). Default is today.
`.DE`	–	yes	End displayed text. Implies `.KE`.
`.DS` *x*	no	yes	Start of displayed text to appear verbatim line–by–line. *x*=I for indented display (default), *x*=L for left-adjusted on the page, *x*=C for centered, *s*=B for make left–justified block, then center whole block. Implies `.KS`.
`.EF` *x*	–	no	Even three-part page footer *x*
`.EN`	–	yes	Space after equation produced by *eqn* or *neqn*.

Table 2-9 -ms *Macro Request Summary— Continued*

Macro Request	Initial Value	Cause Break?	Explanation
.EQ *x y*	–	yes	Precede equation; break out and add space. Equation number is *y*. The optional argument *x* may be I to indent equation (default), L to left–adjust the equation, or C to center it.
.FE	–	yes	End footnote.
.FS *x*	–	no	Start footnote. *x* is optional footnote label. The note will be printed at the bottom of the page.
.I *x*	no	no	Italicize *x*; if *x* is missing, italic text follows.
.IP *x y*	no	yes	Start indented paragraph, with hanging tag *x*. Indentation is *y* ens (default 5).
.KE	–	yes	End keep. Put kept text on next page if not enough room.
.KF	no	yes	Start floating keep. If the kept text must be moved to the next page, float later text back to this page.
.KS	no	yes	Start keeping following text.
.LG	no	yes	Make letters larger.
.LP	yes	yes	Start left-blocked paragraph.
.ND *date*	–	no	Use date supplied if any as page footer; only in special format positions.
.NH *n*	–	yes	Same as .SH with section number supplied automatically. Numbers are multilevel, like 1.2.3, where *n* tells what level is wanted (default is 1).
.NL	yes	no	Make letters normal size.
.IX *x y*	–	yes	Index entries *w* and *y* and so on up to 5 levels. Make letters normal size.
.OF *x*	–	no	Odd three-part page footer.
.OH *header*	–	no	Odd three-part page header.
.P1	–	no	Print header on first page (only in thesis mode).
.PP	no	yes	Begin paragraph. First line indented.
.PT	pg #	–	Page title, automatically invoked at top of page. May be redefined.

Table 2-9 -ms *Macro Request Summary— Continued*

Macro Request	Initial Value	Cause Break?	Explanation
.PX *x*	–	yes	Print table of contents; *x*=no suppresses title.
.QP	–	yes	Begin single paragraph which is indented and shorter.
.R	yes	no	Roman text follows.
.RE	–	yes	End relative indent level.
.RP	no	–	Cover sheet and first page for released paper. Must precede other requests.
.RS	–	yes	Start level of relative indentation. Following .IPs are measured from current indentation.
.SH	–	yes	Section head follows, font automatically bold.
.SM	no	no	Make letters smaller.
.TA *x*...	5...	no	Set tabs in ens. Default is 5 10 15 ...
.TE	–	yes	End table.
.TH	–	yes	End heading section of table.
.TL	no	yes	Title follows.
.TM	off	no	Thesis mode format.
.TS *x*	–	yes	Begin table; if *x* is H, table has repeated heading on subsequent pages.
.UL *x*	–	yes	Underline argument, even in troff.
.XA *xy*	–	yes	Another index entry; *x*=page for no for none, *y*=indent.
.XE	–	yes	End index entry or series of .IX entries.
.XS *xy*	–	yes	Begin index entry; *x*=page or no for none, *y*=indent.
.UL *x*	–	yes	Underline argument, even in troff.

Table 2-10 -ms *String Definitions*

Name	Definition	In nroff	In troff
quote	*Q	"	“
unquote	*U	"	”
dash	*-	--	—
month of year	*(MO	March	March
current date	*(DY	6 March 1988	6 March 1988
numbered footnote	**	[1]*footnote*	[1]*footnote*

The following table summarizes command lines you use to print and display documents. Use the same order with `troff` for preprocessing files with `tbl` and `eqn`.

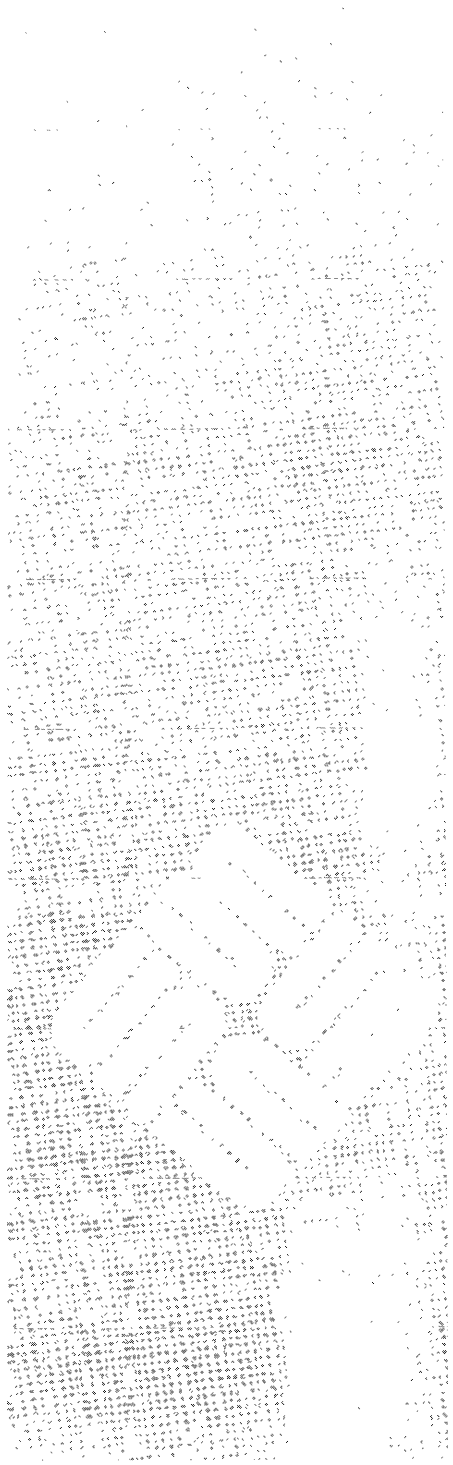
If you use the two-column `.2C` request, either pipe the `nroff` output through `col` or make the first line of the input `.pi /usr/bin/col`.

Table 2-11 *Printing and Displaying Documents*

What You Want to Do	How to Do it
Display file on screen	**nroff -ms** *file(s)* **| more**
Print file on line printer	**nroff -ms** *file(s)* **| lpr**
Print file with tables	**tbl** *file(s)* **| nroff -ms | lpr**
Print file with equations	**neqn** *file(s)* **| nroff -ms | lpr**
Print file with both	**tbl** *file(s)* **| neqn | nroff -ms | lpr**
Print file using `troff`	**troff -ms** *file(s)* **| lpr -t**

3

The -man Macro Package



3

The -man Macro Package

The -man macro package is used to format the manual pages to look like those in the *SunOS Reference Manual*, for example.

3.1. Parts of a Manual Page

A manual page consists of several parts:

- The first part is the *header and footer* or .TH line. This line identifies the manual page and sets up the titles and other information to print the page headers and footers.
- The next few sections are all introduced by .SH macro requests.

A skeleton command file would look something like this:

```
.TH XX 1 "7 November 1984"
.SH NAME
.SH SYNOPSIS
.SH DESCRIPTION
.SH OPTIONS
.SH FILES
.SH "SEE ALSO"
.SH DIAGNOSTICS
.SH BUGS
```

The sections have the following meanings:

NAME	The name of the command and a short description.
SYNOPSIS	A short synopsis of the command including its options and arguments.
DESCRIPTION	A brief narrative description of what the command does.
OPTIONS	A list of the options in terse itemized list format.
FILES	Names of files that this command uses or creates.
SEE ALSO	Other relevant commands and files and manuals.
BUGS	Known deficiencies in the command.

Occasionally there may be other sections you can add. For instance, a couple of the manual pages have a section called RESTRICTIONS, which contains the notice that this software is not distributed outside of the United States of America.

Leave out sections that do not apply — it is not necessary to have a title without any content to go with it. Definitely avoid sections that read:

```
BUGS
    None.
```

3.2. Coding Conventions

The following subsections compose a fairly detailed description of what the different sections of the manual page contain.

The Header and Footer Line (.TH) — Identifying the Page

The `.TH` macro is the macro that identifies the page. The format is

```
.TH n c x v m
```

This means, for example: Begin page named *n* of chapter *c*. The *x* argument is for extra commentary for the center page footer. The *v* argument alters the left portion of the page footer. The *m* argument alters the center portion of the page header. The `.TH` command line also incidentally sets the prevailing indent and tabs to .5i.

To code a manual page called `troff(1)`, for example, you would code a `.TH` macro like:

```
.TH TROFF 1 "today's date"
```

The third parameter to the `.TH` macro is the date on which you created or last changed the manual page. You code *today's date* in international form:

numerical day spelled-out month numerical year

So if today is September 3rd, 1984, you code the **`.TH`** macro like:

```
.TH TROFF 1 "3 September 1984"
```

The NAME Line

The **NAME** line is a one-liner that identifies the command or program. You code the information like this:

```
.SH NAME
troff \- typeset or format documents
```

This line must be typed all in the Roman font with no font changes or point-size changes or any other text manipulation. Typing the command line all in Roman with no text manipulation is for the permuted index generator. It gets all confused if there is anything in that line other than plain text.

Note the \- in there — why do we type a \-? Well, in `troff` jargon, a simple - sign gets you a hyphen. We actually would like a en-dash (like –) instead of a hyphen, in lieu of actually having a em-dash (like —). This use of the \- to get a – is a UNIX† tradition.

The SYNOPSIS Section

The **SYNOPSIS** line(s) show the user what options and arguments can be typed. The conventions for the `SYNOPSIS` have varied wildly over the years. Nonetheless, here are the guidelines:

- *Literal text* (that is, what the user types) is coded in **boldface.**
- *Variables* (that is, things someone might *substitute for*) are typed in *italic text.*
- *Optional things* are enclosed in brackets — that is the characters `[` and `]`.
- *Alternatives* are separated by the vertical bar sign (|).

The synopsis should show what the options are — some manual pages used to read like this:

SYNOPSIS
troff [*options*] filename ...

but it *should* read:

SYNOPSIS
troff [**–o***pagelist*] [**–n***N*] [**–m** *name*] ... [filename]

The DESCRIPTION Section

The **DESCRIPTION** section of a manual page should contain a brief description of what the command does *for the user*, in terms that the user cares about.

Within the `DESCRIPTION` and `OPTIONS` sections, *italic text* is used for filenames and command names. The rationale here is that UNIX commands are simply files. When referring to other manual pages, you type the name in italics and the following parenthesized section number in Roman, as in `make(1)`. Use the `-man` macro `.IR` to get alternating words joined in italic and Roman fonts. Note that the macros that join alternating words in different fonts (`.IR`, `.IB`, `.BR`, `.BI`, `.RI`, `.RB`) all accept only *six* parameters. See the section on how to format a manual page for more formatting rules.

Part of the description in the `grep` manual page used to read:

..... `grep` patterns are limited regular expressions in the style of `ed(1)`; it uses a compact nondeterministic algorithm. `egrep` patterns are full regular expressions; it uses a fast deterministic algorithm that sometimes needs exponential space. `fgrep` patterns are fixed strings; it is fast and compact.

Most users do not care that `egrep` uses a fast deterministic algorithm. As an example of a more useful way of describing a command for the user, here is how

† UNIX is a registered trademark of AT&T.

that sentence in the `grep` manual page currently reads.

..... `grep` patterns are limited regular expressions in the style of `ed(1)`. `egrep` patterns are full regular expressions including alternation. `fgrep` searches for lines that contain one of the (newline-separated) *strings*. `fgrep` patterns are fixed strings — no regular expression metacharacters are supported.

Here's another bad example: the `lpr(1)` command used to tell you that the **–s** option uses the `symlink` (2) system call to make a symbolic link to the data file instead of copying the data file to the spool area. The user may not know what this means or how to use the information. The description was changed to just tell you that the **–s** option makes a symbolic link to the data file. How it is done is of little concern to some poor blighter who just wants to print a file.

The OPTIONS Section

The **OPTIONS** section of a manual page contains an itemized list of the options that the command recognizes, and how the options affect the behavior of the command. The general format for this section is

–option Description of what the option does.

A specific example from the `troff` manual page looks like this:

OPTIONS
Options may appear in any order as long as they appear **before** the files.

–o*list*
Print only pages whose page numbers appear in the comma-separated list of numbers and ranges. A range N-M 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.

–n*N*
Number first generated page *N*.

–m*name*
Prepend the macro file /usr/lib/tmac/tmac.name to the input files.

–r*aN*
Set register *a* (one-character) to *N*.

–i
Read standard input after the input files are exhausted.

–q
Invoke the simultaneous input-output mode of the *rd* request.

–t
Direct output to the standard output instead of the phototypesetter. In general, you will have to use this option if you don't have a typesetter attached to the system.

–a

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

Some options of `troff` only apply if you have a C/A/T typesetter attached to your system. These options are here for historical reasons:

–s*N*
Stop every *N* pages. `troff` stops the phototypesetter every *N* pages, produces a trailer to allow changing cassettes, and resumes when the typesetter's start button is pressed.

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

–w
Wait until phototypesetter is available, if currently busy.

-b
Report whether the phototypesetter is busy or available. No text processing is done.

–p*N*
Print all characters in point size *N* while retaining all prescribed spacings and motions, to reduce phototypesetter elapsed time.

The FILES Section

The **FILES** section of a manual page contains a list of the files that the program accesses, creates, or modifies. Obviously, you can leave this section out if the program uses no files.

The example from the `troff` manual page looks like this:

> If the file `/usr/adm/tracct` is writable, `troff` keeps phototypesetter accounting records there. The integrity of that file may be secured by making `troff` a 'set user-id' program.

FILES

/tmp/ta*	temporary file
/usr/lib/tmac/tmac.*	standard macro files
/usr/lib/term/*	terminal driving tables for nroff
/usr/lib/font/*	font width tables for troff
/dev/cat	phototypesetter
/usr/adm/tracct	accounting statistics for /dev/cat

The SEE ALSO Section

The **SEE ALSO** section of a manual page contains a list of references to other programs, files, and manuals relating to this program. For example, on the `troff` manual page, the **SEE ALSO** section looks like this:

> SEE ALSO
> *Formatting Documents* and *Using NROFF and TROFF*,
> `nroff(1), eqn(1), tbl(1), ms(7), me(7),`
> `man(7), col(1)`

Make sure that the references are useful — the `rm(1)` command references the `unlink(2)` system call. Does the user care what system call is used to get rid of a file? It's not intuitive that you use a function called `unlink` to remove a file.

Leave this section out if there are no interesting references.

The BUGS Section

The **BUGS** section of a manual page is to convey limitations or bad behavior of the command to the reader. Please limit bugs to these categories.

Leave this section out altogether if there are no bugs worth noting.

3.3. New Features of the -man Macro Package

New Number Registers

Recent enhancements to the `-man` macro package facilitate including manual pages in manuals. The major new features are number registers that can be set from the `itroff`, `iroff`, `troff`, `ditroff`, or `nroff` command line. The number registers are:

D	Format the document for double-sided printing if the **D** number register is set to 1. Double-sided printing means that the page numbers appear in different locations on odd and even pages. Page numbers appear in the running footers in the lower *right* corner of *odd-numbered* pages and in the lower *left* corner of *even-numbered* pages.
C	Number pages contiguously — pages are numbered 1, 2, 3,... even when you format more than one manual page at a time. Every new topic used to start numbering at page 1.
P*nnn*	Start Page numbering at page *nnn* — page numbering starts at page 1 if not otherwise specified.
X*nnn*	Number pages as *nnn***a**, *nnn***b**, etc when the current page number becomes *nnn*. This feature is for generating update pages to slot in between existing pages. For example, if a new page called *skyversion*(8) should be included in an interim release, we can number that page as page '26a' and drop it into the existing manual in a reasonable fashion.

Using the Number Registers

Number registers are set from the `itroff`, `iroff`, `troff`, `ditroff`, or `nroff` command line by the **–r** (set register) option, followed immediately by the *one-letter* name of the register, followed immediately by the value to put into the number register:

```
hostname% troff -man -rD1 manpage.1
hostname%
```

This example shows how to request a format suitable for double-sided printing.

If your `grab(1)` manual page used to be three pages long and is now five pages long, you need the pages numbered 1, 2, 3, 3a, and 3b instead of 1, 2, 3, 4, and 5. You get this effect by using the **–rX** option on the command line, setting the **X** register to 3:

```
hostname% troff -man -rX3 grab.1
hostname%
```

We introduced the `screendump(1)` and `screenload(1)` manual pages in the 1.2 release. `screendump(1)` and `screenload(1)` come immediately after the `sccsdiff(1)` manual page. `sccsdiff` 's last page number is page 260, so we get `screendump(1)` and `screenload(1)` formatted with this command to start page numbering at 260 and to start putting in extra page letters at 260 as well:

```
hostname% troff -man -rP260 -rX260 screendump.1 screenload.1
hostname%
```

3.4. How to Format a Manual Page

Any text argument *t* to a macro request may be from zero to six words. Quotes my be used to include blanks in a 'word'. If the text field is empty, the macro request is applied to the next input line with text to be printed. In this way, **.I** italicizes an entire line, and **.SM** followed on a separate line by **.B** creates small, bold letters.

A prevailing indent distance is remembered between successive indented paragraphs, and is reset to the default value upon reaching a non-indented paragraph. Default units for indents *i* are ens.

Type font and size are reset to the default values before each paragraph, and after processing font- and size-setting macros.

These strings are predefined by `-man`:

*R ®, '(Reg)' in `nroff`.
*S Change to default type size.

Table 3-1 *Summary of the* -man *Macro Requests*

Request	*Cause Break*	*If no Argument*	*Explanation*
.B *t*	no	*t*=next text line	Text *t* is bold.
.BI *t*	no	*t*=next text line	Join words of *t* alternating bold and italic.
.BR *t*	no	*t*=next text line	Join words of *t* alternating bold and Roman.
.DT	no	.5i 1i...	Restore default tabs.
.HP *i*	yes	*i*=prevailing indent	Set prevailing indent to *i*. Begin paragraph with hanging indent.
.I *t*	no	*t*=next text line	Text *t* is italic.
.IB *t*	no	*t*=next text line	Join words of *t*alternating italic and bold.
.IP *x i*	yes	*x*=""	Same as .TP with tag *x*.
.IR *t*	no	*t*=next text line	Join words of *t* alternating italic and Roman.
.LP	yes		Same as .PP.
.PD *d*	no	*d*=.4v	Interparagraph distance is *d*.
.PP	yes		Begin paragraph. Set prevailing indent to .5i.
.RE	yes		End of relative indent. Set prevailing indent to amount of starting .RS.
.RB *t*	no	*t*=next text line	Join words of *t* alternating Roman and bold.
.RI *t*	no	*t*=next text line	Join words of *t* alternating Roman and italic.
.RS *i*	yes	*i*=prevailing indent	Start relative indent, move left margin in distance *i*. Set prevailing indent to .5i for nested indents.
.SB *t*	no		Print *t* in smaller boldface.
.SH *t*	yes	*t*=next text line	Subheading.
.SM *t*	yes	*t*=next text line	Text *t* is two point sizes smaller than surrounding text.
.TH *n c x v m*	yes		Begin page named *n* of chapter *c*. The *x* argument is for extra commentary for the center page footer. The *v* argument alters the left portion of the page footer. The *m* argument alters the center portion of the page header. The .TH command line also incidentally sets the prevailing indent and tabs to .5i.
.TP *i*	yes	*i*=prevailing indent	Set the prevailing indent to *i*. Begin indented paragraph with hanging tag given by the next text line. If the tag does not fit, place it on a separate line.
.TX *t p*	no		Resolve the title abbreviation *t*; join to punctuation *p*.

To learn how to format manual pages on your terminal or workstation screen, refer to the man(1) manual page.

4

Formatting Documents with the -me Macros

4

Formatting Documents with the -me Macros

This chapter describes the -me macro package.[1] The first part of each section presents the material in user's guide format and the second part lists the macro requests for quick reference. The chapter contents include descriptions of the basic requests, displays, annotations, such as footnotes, and how to use -me with nroff and troff.

We assume that you are somewhat familiar with nroff and troff and that you know something about breaks, fonts, point sizes, the use and definition of number registers and strings, and scaling factors for ens, points, vertical line spaces, etc. If you are a newcomer, try out the basic features as you read along.

All request names in -me follow a naming convention. You may define number registers, strings, and macros, provided that you use single-character, upper case names or double-character names consisting of letters and digits with at least one upper case letter. Do not use special characters in the names you define. The word *argument* in this chapter means a word or number which appears on the same line as a request and which modifies the meaning of that request. Default parameter values are given in brackets. For example, the request

```
.sp
```

spaces one line, and

```
.sp 4
```

spaces four lines. The number '4' is an *argument* to the .sp request; it modifies .sp to produce four lines instead of one. Spaces separate arguments from the request and from each other.

[1] The material in this chapter is derived from *Writing Papers with* nroff *Using* -me, E. P. Allman, and -me *Reference Manual*, E. P. Allman, University of California, Berkeley.

4.1. Using -me

When you have your raw text ready, run the `nroff` formatter with the `-me` option to send the output to the standard output, your workstation screen. Type:

```
hostname% nroff -me -Ttype files
hostname%
```

where *type* describes the type of terminal you are outputting to. Common values are `dtc` for a DTC 300s (daisy-wheel type) printer and `lpr` for the line printer. If you omit the `-T` flag, a 'lowest common denominator' terminal is assumed; this is good for previewing output on most terminals.

For easier viewing, pipe the output to `more` or redirect it to another file.

For formatting on the phototypesetter with `troff` (or your installation's equivalent), use:

```
hostname% troff -me file
hostname%
```

4.2. Basic -me Requests

The following sections provide descriptions and examples of the basic `-me` requests.

Paragraphs

The `-me` package has requests for formatting standard, left block, and indented paragraphs.

Standard Paragraph — .pp

Begin standard paragraphs by using the `.pp` request. For example, the input:

```
.pp
Now is the time for all good men
to come to the aid of their party.
Four score and seven years ago,...
```

produces

Now is the time for all good men to come to the aid of their party. Four score and seven years ago,...

that is, a blank line followed by an indented first line.

Do not begin the sentences of a paragraph with a space, since blank lines and lines beginning with spaces cause a break. For example, if you type:

```
.pp
Now is the time for all good men
      to come to the aid of their party.
Four score and seven years ago,...
```

The output is:

Now is the time for all good men
to come to the aid of their party. Four score and seven years ago,...

A new line begins after the word 'men' because the second line begins with a space character.

Because the first call to one of the paragraph macros defined in a section or the `.H` macro *initializes* the macro processor, do not use any of the following requests: `.sc`, `.lo`, `.th`, or `.ac` (see the section called "Section Headings"). Also, avoid changing parameters, notably page length and header and footer margins, which have a global effect on the format of the page.

Left Block Paragraphs — `.lp`

A formatted paragraph can start with a blank line and with the first line indented. You can get left-justified block-style paragraphs as shown throughout this manual by using `.lp` (left paragraph) instead of `.pp`.

Indented Paragraphs — `.ip` and `.np`

Sometimes you want to use paragraphs that have the *body* indented, and the first line exdented, that is, the opposite of indented, with a label. Use the `.ip` request for this. A word specified on the same line as `.ip` is printed in the margin, and the body is lined up at a specified position. For example, the input:

```
.ip one
This is the first paragraph.
Notice how the first line
of the resulting paragraph lines up
with the other lines in the paragraph.
.ip two
And here we are at the second paragraph already.
You may notice that the argument to .ip
appears in the margin.
.lp
We can continue text...
```

produces as output:

one This is the first paragraph. Notice how the first line of the resulting paragraph lines up with the other lines in the paragraph.

two And here we are at the second paragraph already. You may notice that the argument to `.ip` appears in the margin.

We can continue text without starting a new indented paragraph by using the `.lp` request.

If you have spaces in the label of an `.ip` request, use an *unpaddable space* instead of a regular space. This is typed as a backslash character (\) followed by a space. For example, to print the label 'Part 1', type:

```
.ip "Part\ 1"
```

If a label of an indented paragraph, that is, the argument to `.ip`, is longer than the space allocated for the label, `.ip` begins a new line after the label. For

example, the input:

```
.ip longlabel
This paragraph has a long label.
The first character of text on the first line will not
line up with the text on second and subsequent lines,
although they will line up with each other.
```

produces:

longlabel
This paragraph has a long label. The first character of text on the first line will not line up with the text on second and subsequent lines, although they will line up with each other.

You can change the size of the label by using a second argument which is the size of the label. For example, you can produce the above example correctly by saying:

```
.ip longlabel 10
```

which will make the paragraph indent 10 spaces for this paragraph only. For example:

longlabel
This paragraph has a long label. The first character of text on the first line will not line up with the text on second and subsequent lines, although they will line up with each other.

If you have many paragraphs to indent all the same amount, use the *number register* `ii`. For example, to leave one inch of space for the label, type:

```
.nr ii 1i
```

somewhere before the first call to `.ip`.

If you use `.ip` without an argument, no hanging tag is printed. For example, the input:

```
.ip [a]
This is the first paragraph of the example.
We have seen this sort of example before.
.ip
This paragraph is lined up with the previous paragraph,
but it does not have a tag in the margin.
```

produces as output:

[a] This is the first paragraph of the example. We have seen this sort of example before.

This paragraph is lined up with the previous paragraph, but it does not have a tag in the margin.

A special case of `.ip` is `.np`, which automatically numbers paragraphs sequentially from 1. The numbering is reset at the next `.pp`, `.lp`, or `.H` request. For example, the input:

```
.np
This is the first point.
.np
This is the second point.
Points are just regular paragraphs
which are given sequence numbers automatically
by the .np request.
.lp
This paragraph will reset numbering by .np.
.np
For example,
we have reverted to numbering from one now.
```

generates:

(1) This is the first point.

(2) This is the second point. Points are just regular paragraphs which are given sequence numbers automatically by the `.np` request.

This paragraph will reset numbering by `.np`.

(1) For example, we have reverted to numbering from one now.

Paragraph Reference

`.lp` Begin left-justified paragraph. Centering and underlining are turned off if they were on, the font is set to `\n(pf` [1], the type size is set to `\n(pp` [10p], and a `\(nps` space is inserted before the paragraph (0.35v in `troff`, 1v or 0.5v in `nroff` depending on device resolution). The indent is reset to `\n($1` [0] plus `\n(po` [0] unless the paragraph is inside a display (see `.ba` in "Miscellaneous Requests"). At least the first two lines of the paragraph are kept together on a page.

`.pp` Like `.lp`, except that it puts `\n(pi` [5n] units of indent. This is the standard paragraph macro.

`.ip` *T I* Indented paragraph with hanging tag. The body of the following paragraph is indented *I* spaces (or `\n(ii` [5n] spaces if *I* is not specified) more than a non-indented paragraph is (such as with `.lp`). The title *T* is exdented. The result is a paragraph with an even left edge and *T* printed in the margin. Any spaces in *T* must be unpaddable. If *T* will not fit in the space provided, `.ip` starts a new line.

`.np` An `.ip` variant that numbers paragraphs. Numbering is reset after an `.lp`, `.pp`, or `.H`. The current paragraph number is in `\n$p`.

4.3. Headers and Footers — `.he` and `.fo`

You can put arbitrary headers and footers at the top and bottom of every page. Two requests of the form `.he` *title* and `.fo` *title'* define the titles to put at the head and the foot of every page, respectively. The titles are called *three-part* titles, that is, there is a left-justified part, a centered part, and a right-justified part. The first character of *title* (whatever it may be) is used as a delimiter to separate these three parts. You can use any character but avoid the backslash and double quote marks. The percent sign is replaced by the current page number whenever it is found in the title. For example, the input:

```
.he '' %''
.fo 'Jane Jones'' My Book'
```

results in the page number centered at the top of each page, 'Jane Jones' in the lower left corner, and 'My Book' in the lower right corner.

If there are two blanks adjacent anywhere in the title or more than eight blanks total, you must enclose three-part titles in single quotes.

Headers and footers are set in font `\n(tf` [3] and size `\n(tp` [10p]. Each of the definitions applies as of the *next* page.

Three number registers control the spacing of headers and footers. `\n(hm` [4v] is the distance from the top of the page to the top of the header, `\n(fm` [3v] is the distance from the bottom of the page to the bottom of the footer, `\n(tm` [7v] is the distance from the top of the page to the top of the text, and `\n(bm` [6v] is the distance from the bottom of the page to the bottom of the text (nominal). You can also specify the space between the top of the page and the header, the header and the first line of text, the bottom of the text and the footer, and the footer and the bottom of the page with the macros `.m1`, `.m2`, `.m3`, and `.m4`.

Headers and Footers Reference

`.he` *'l'm'r'* Define three-part header, to be printed on the top of every page.

`.fo` *'l'm'r'* Define footer, to be printed at the bottom of every page.

`.eh` *'l'm'r'* Define header, to be printed at the top of every even-numbered page.

`.oh` *'l'm'r'* Define header, to be printed at the top of every odd-numbered page.

`.ef` *'l'm'r'* Define footer, to be printed at the bottom of every even-numbered page.

`.of` *'l'm'r'* Define footer, to be printed at the bottom of every odd-numbered page.

`.hx` Suppress headers and footers on the next page.

`.m1 +`*N* Set the space between the top of the page and the header [4v].

`.m2 +N`	Set the space between the header and the first line of text [2v].
`.m3 +N`	Set the space between the bottom of the text and the footer [2v].
`.m4 +N`	Set the space between the footer and the bottom of the page [4v].
`.ep`	End this page, but do not begin the next page. Useful for forcing out footnotes. Must be followed by a `.bp` or the end of input.
`.$h`	Called at every page to print the header. May be redefined to provide fancy headers, such as, multi-line, but doing so loses the function of the `.he`, `.fo`, `.eh`, `.oh`, `.ef`, and `.of` requests, as well as the chapter-style title feature of `.+c`.
`.$f`	Print footer; same comments apply as in `.$h`.
`.$H`	A normally undefined macro which is called at the top of each page after processing the header, initial saved floating keeps, etc.; in other words, this macro is called immediately before printing text on a page. Used for column headings and the like.

Double Spacing — `.ls 2`

`nroff` will double space output text automatically if you use the request `.ls 2`, as is done in this section. You can revert to single-space mode by typing `.ls 1`.

Page Layout

You can change the way the printed copy looks, sometimes called the *layout* of the output page with the following requests. Most of these requests adjust the placing of 'white space' (blank lines or spaces). In these explanations, replace characters in italics with values you wish to use; bold characters represent characters which you should actually type.

Use `.bp` (break page) to start a new page.

The request `.sp` *N* leaves *N* lines of blank space. You can omit *N* to skip a single line or you can use the form *N* `i` (for *N* inches) or *N* `c` (for *N* centimeters). For example, the input:

```
.sp 1.5i
My thoughts on the subject
.sp
```

leaves one and a half inches of space, followed by the line 'My thoughts on the subject', followed by a single blank line.

The `.in` +*N* (indent) request changes the amount of white space on the left of the page. The argument *N* can be of the form + *N* (meaning leave *N* spaces more than you are already leaving), – *N*' (meaning leave *N* spaces less than you do now), or just *N* (meaning leave exactly *N* spaces). *N* can be of the form *N* `i` or *N* `c` also. For example, the input:

```
initial text
.in 5
some text
.in +1i
more text
.in -2c
final text
```

produces 'some text' indented exactly five spaces from the left margin, 'more text' indented five spaces plus one inch from the left margin (fifteen spaces on a pica typewriter), and 'final text' indented five spaces plus one inch minus two centimeters from the margin. That is, the output is:

initial text
some text
more text
final text

The `.ti +N` (temporary indent) request is used like `.in +N` when the indent should apply to one line only, after which it should revert to the previous indent. For example, the input:

```
.in 1i
.ti 0
Ware, James R.  The Best of Confucius,
Halcyon House, 1950.
An excellent book containing translations of
most of Confucius' most delightful sayings.
A definite must for anyone interested in the
early foundations of Chinese philosophy.
```

produces:

Ware, James R. The Best of Confucius, Halcyon House, 1950. An excellent book containing translations of most of Confucius' most delightful sayings. A definite must for anyone interested in the early foundations of Chinese philosophy.

You can center text lines with the `.ce` (center) request. The line after the `.ce` is centered horizontally on the page. To center more than one line, use `.ce` *N*, where *N* is the number of lines to center, followed by the *N* lines. If you want to center many lines but don't want to count them, type:

```
.ce 1000
lines to center
.ce 0
```

The `.ce 0` request tells `nroff` to center zero more lines, in other words, to stop centering.

All of these requests cause a break; that is, they always start a new line. If you want to start a new line without performing any other action, use `.br` (break).

Underlining — `.ul`

Use the `.ul` (underline) request to underline text. The `.ul` request operates on the next input line when it is processed. You can underline multiple lines by stating a count of *input* lines to underline, followed by those lines, the same as with the `.ce` request. For example, the input:

```
.ul 2
The quick brown fox
jumped over the lazy dog.
```

underlines those words in `nroff`. In `troff` they are italicized.

Displays

Use displays to set off sections of text from the body of the paper. Major quotes, tables, and figures are types of displays, as are all the examples used in this manual. All displays except centered text blocks are single-spaced.

Major Quotes — `.(q` and `.)q`

Major quotes are quotes which are several lines long, and hence are set in from the rest of the text without quote marks around them. Use `.(q` and `.)q` to surround the quote. For example, the input:

```
As Weizenbaum points out:
.(q
It is said that to explain is to explain away.
This maxim is nowhere so well fulfilled
as in the areas of computer programming,...
.)q
```

generates as output:

As Weizenbaum points out:

> It is said that to explain is to explain away. This maxim is nowhere so well fulfilled as in the areas of computer programming,...

Lists — `.(l` and `.)l`

A *list* is an indented, single-spaced, unfilled display. You should use lists when the material to be printed should not be filled and justified like normal text. This is useful for columns of figures, for example. Surround the list text by the requests `.(l` and `.)l`. For example, type:

```
Alternatives to avoid deadlock are:
.(l
Lock in a specified order
Detect deadlock and back out one process
Lock all resources needed before proceeding
.)l
```

to produce:

Alternatives to avoid deadlock are:

Lock in a specified order
Detect deadlock and back out one process
Lock all resources needed before proceeding

Keeps — `.(b` and `.)b`, `.(z` and `.)z`

A *keep* is a group of lines that are kept together on a single page. If less vertical space exists on the current page than can accommodate text within a keep, the formatter begins a new page and keeps those lines together. Keeps are useful for printing diagrams because you don't want them spread across two pages. For comparison, lists may be broken over a page boundary, whereas keeps may not.

Blocks are the basic kind of keep. They begin with the request `.(b` and end with the request `.)b`. If there is not enough room on the current page for everything in the block, the formatter begins a new page. This has the unaesthetic effect of leaving blank space at the bottom of the page. When this is not appropriate, you can use the alternative called a *floating keep.*

Floating keeps move relative to the text. Hence, they are good for things which will be referred to by name, such as 'See figure 3'. A floating keep will appear at the bottom of the current page if it will fit; otherwise, it will appear at the top of the next page. Floating keeps begin with the line `.(z` and end with the line `.)z`. An example of a floating keep is:

```
.(z
.hl
Text of keep to be floated.
.sp
.ce
Figure 1.  Example of a Floating Keep.
.hl
.)z
```

The `.hl` request draws a horizontal line so the figure stands out from the text.

4.4. Fancy Displays

Keeps and lists are normally collected in *nofill* mode, so they are good for tables and displays. If you want a display in fill mode (for text), type `.(l F`. Throughout this section, comments applied to `.(l` also apply to `.(b` and `.(z`. This kind of display produced by `.(l` is indented from both margins. For example, the input:

```
.(l F
And now boys and girls,
a newer, bigger, better toy than ever before!
Be the first on your block to have your own computer!
Yes kids, you too can have one of these modern
data processing devices.
You too can produce beautifully formatted papers
without even batting an eye!
.)l
```

will be formatted as:

> And now boys and girls, a newer, bigger, better toy than ever before! Be the first on your block to have your own computer! Yes kids, you too can have one of these modern data processing devices. You too can produce beautifully formatted papers without even batting an eye!

Lists and blocks are also normally indented, while floating keeps are normally left-justified. To get a left-justified list, type `.(l L`. To center a list line-for-line, type `.(l C`. For example, to get a filled, left-justified list, use:

```
.(l L F
text of block
.)l
```

The input:

```
.(l
first line of unfilled display
more lines
.)l
```

produces the indented text:

> first line of unfilled display
> more lines

Typing the character `L` after the `.(l` request produces the left-justified result:

first line of unfilled display
more lines

Using `C` instead of `L` produces the line-at-a-time centered output:

first line of unfilled display
more lines

Sometimes you may want to center several lines as a group, rather than centering them one line at a time. To do this use centered blocks, which are surrounded by the requests `.(c` and `.)c`. All the lines are centered as a unit, such that the longest line is centered, and the rest are lined up around that line. Notice that lines do not move relative to each other using centered blocks, whereas they do using the `C` keep argument.

Centered blocks are *not* keeps, and you may use them in conjunction with keeps. For example, to center a group of lines as a unit and keep them on one page, use:

```
.(b L
.(c
first line of unfilled display
more lines
.)c
.)b
```

to produce:

first line of unfilled display
more lines

the result would have been the same, but with no guarantee that the lines of the centered block would have all been on one page. Note the use of the `L` argument to `.(b`; this centers the centered block within the entire line rather than within the line minus the indent. Also, you must nest the center requests *inside* the keep requests.

Display Reference

All displays except centered blocks and block quotes are preceded and followed by an extra `\n(bs` (same as `\n(ps`) space. Quote spacing is stored in a separate register; centered blocks have no default initial or trailing space. The vertical spacing of all displays except quotes and centered blocks is stored in register `\n($R` instead of `\n($r`.

`.(l` *mf* Begin list. Lists are single-spaced, unfilled text. If *f* is `F`, the list will be filled. If *m* [`I`] is `I` the list is indented by `\n(bi` [4n]; if it is `M`, the list is indented to the left margin; if it is `L`, the list is left-justified with respect to the text (different from `M` only if the base indent (stored in `\n($i` and set with `.ba`) is not zero); and if it is `C`, the list is centered on a line-by-line basis. The list is set in font `\n(df` [0]. You must use a matching `.)l` to end the list. This macro is almost like `.DS` except that no attempt is made to keep the display on one page.

`.)l` End list.

`.(q` Begin major quote. The lines are single-spaced, filled, moved in from the main body of text on both sides by `\n(qi` [4n], preceded and followed by `\n(qs` (same as `\n(bs`) space, and are set in point size `\n(qp`, that is, one point smaller than the surrounding text.

`.)q` End major quote.

`.(b` *mf* Begin block. Blocks are a form of *keep*, where the text of a keep is kept together on one page if possible. Keeps are useful for tables and figures which should not be broken over a page. If the block will not fit on the current page a new page is begun, unless that would leave more than `\n(bt` [0] white space at the bottom of the text. If `\n(bt` is zero, the threshold feature is turned off. Blocks are not filled unless *f* is `F`, when they are filled. The block will be left-justified if *m* is `L`, indented by `\n(bi` [4n] if *m* is `I` or absent,

centered (line-for-line) if *m* is `C`, and left justified to the margin, not to the base indent, if *m* is `M`. The block is set in font `\n(df` [0].

`.)b` End block.

`.(z` *mf* Begin floating keep. Like `.(b` except that the keep is *floated* to the bottom of the page or the top of the next page. Therefore, its position relative to the text changes. The floating keep is preceded and followed by `\n(zs` [1v] space. Also, it defaults to mode `M`.

`.)z` End floating keep.

`.(c` Begin centered block. The next keep is centered as a block, rather than on a line-by-line basis as with `.(b C`. This call may be nested inside keeps.

`.)c` End centered block.

Annotations

There are a number of requests to save text for later printing. *Footnotes* are printed at the bottom of the current page. *Delayed text* is intended to be a variant form of footnote; the text is printed only when explicitly called for, such as at the end of each chapter. *Indexes* are a type of delayed text having a tag, usually the page number, attached to each entry after a row of dots. Indexes are also saved until explicitly called for.

Footnotes — `.(f` and `.)f`

Footnotes begin with the request `.(f` and end with the request `.)f`. The current footnote number is maintained automatically, and can be used by typing `\**`, to produce a footnote number.[2] The number is automatically incremented after every footnote. For example, the input:

```
.(q
A man who is not upright
and at the same time is presumptuous;
one who is not diligent and at the same time is ignorant;
one who is untruthful and at the same time is incompetent;
such men I do not count among acquaintances.\**
.(f
\**James R. Ware,
.ul
The Best of Confucius,
Halcyon House, 1950.
Page 77.
.)f
.)q
```

generates the result:

> A man who is not upright and at the same time is presumptuous; one who is not diligent and at the same time is ignorant; one who is untruthful and at the same time is incompetent; such men I do not count among acquaintances.[3]

Make sure that the footnote appears *inside* the quote, so that the footnote will appear on the same page as the quote.

Delayed Text

Delayed text is very similar to a footnote except that it is printed when explicitly called for. Use this feature to put a list of references at the end of each chapter, as is the convention in some disciplines. Use `\*#` on delayed text instead of `\**` as on footnotes.

If you are using delayed text as your standard reference mechanism, you can still use footnotes, except that you may want to refer to them with special characters* rather than numbers.

Indexes — `.(x` `.)x` and `.xp`

An *index* resembles delayed text, in that it is saved until called for. It is actually more like a table of contents, since the entries are not sorted alphabetically. However, each entry has the page number or some other tag appended to the last line of the index entry after a row of dots.

Index entries begin with the request `.(x` and end with `.)x`. An argument to the `.)x` indicates the value to print as the 'page number.' It defaults to the current page number. If the page number given is an underscore (_), no page number or

---

[2] Like this.

[3] James R. Ware, *The Best of Confucius*, Halcyon House, 1950. Page 77.

* Such as an asterisk.

line of dots is printed at all. To get the line of dots without a page number, type **.)x ""**, which specifies an explicitly null page number.

The .xp request prints the index.

For example, the input:

```
.(x
Sealing wax
.)x 9
.(x
Cabbages and kings
.xp
```

generates:

Sealing wax 9

Cabbages and kings

< etc. >

The .(x request may have a single-character argument, specifying the *name* of the index; the normal index is **x**. Thus, you can maintain several *indices* simultaneously, such as a list of tables and a table of contents.

Notice that the index must be printed at the *end* of the paper, rather than at the beginning where it will probably appear (as a table of contents); you may have to rearrange the pages after printing.

Annotations Reference

.(d	Begin delayed text. Everything in the next keep is saved for output later with .pd in a manner similar to footnotes.
.)d *n*	End delayed text. The delayed text number register \n($d and the associated string *# are incremented if *# has been referenced.
.pd	Print delayed text. Everything diverted via .(d is printed and truncated. You might use this at the end of each chapter.
.(f	Begin footnote. The text of the footnote is floated to the bottom of the page and set in font \n(ff [1] and size \n(fp [8p]. Each entry is preceded by \n(fs [0.2v] space, is indented \n(fi [3n] on the first line, and is indented \n(fu [0] from the right margin. Footnotes line up underneath two-column output. If the text of the footnote will not all fit on one page, it will be carried over to the next page.
.)f *n*	End footnote. The number register \n($f and the associated string ** are incremented if they have been referenced.
.$s	The macro to generate the footnote separator. You may redefine this macro to give other size lines or other types of separators. It currently draws a 1.5-inch line.

`.(x` *x* Begin index entry. Index entries are saved in the index *x* until called up with `.xp`. Each entry is preceded by a `\n(xs` [0.2v] space. Each entry is 'undented' by `\n(xu` [0.5i]; this register tells how far the page number extends into the right margin.

`.)x` *P A* End index entry. The index entry is finished with a row of dots with *A* [null] right justified on the last line, such as for an author's name, followed by *P* [`\n%`]. If *A* is specified, *P* must be specified; `\n%` can be used to print the current page number. If *P* is an underscore, no page number and no row of dots are printed.

`.xp` *x* Print index *x* [`x`]. The index is formatted in the font, size, and so forth in effect at the time it is printed, rather than at the time it is collected.

4.5. Fancy Features

A large number of fancier requests exist, notably requests to provide other sorts of paragraphs, numbered sections of the form '1.2.3', such as those used in this manual, and multicolumn output.

Section Headings — `.sh` and `.uh`

You can automatically generate section numbers, using the `.sh` request. You must tell `.sh` the *depth* of the section number and a section title. The depth specifies how many numbers separated by decimal points are to appear in the section number. For example, the section number '4.2.5' has a depth of three.

Section numbers are incremented if you add a number. Hence, you increase the depth, and the new number starts out at one. If you subtract section numbers, or keep the same number, the final number is incremented. For example, the input:

```
.sh 1 "The Preprocessor"
.sh 2 "Basic Concepts"
.sh 2 "Control Inputs"
.sh 3
.sh 3
.sh 1 "Code Generation"
.sh 3
```

produces as output the result:

1. The Preprocessor
1.1. Basic Concepts
1.2. Control Inputs
1.2.1.
1.2.2.
2. Code Generation
2.1.1.

You can specify the beginning section number by placing the section number after the section title, using spaces instead of dots. For example, the request:

```
.sh 3 "Another section" 7 3 4
```

will begin the section numbered '7.3.4'; all subsequent `.sh` requests will be numbered relative to this number.

There are more complex features which indent each section proportionally to the depth of the section. For example, if you type:

```
.nr si Nx
```

each section will be indented by an amount *N*. *N* must have a scaling factor attached, that is, it must be of the form *Nx*, where *x* is a character telling what units *N* is in. Common values for *x* are 'i' for inches, 'c' for centimeters, and 'n' for 'ens,' the width of a single character. For example, to indent each section one-half inch, type:

```
.nr si 0.5i
```

The request indents sections by one-half inch per level of depth in the section number. As another example, consider:

```
.nr si 3n
```

which gives three spaces of indent per section depth.

You can produce section headers without automatically generated numbers using:

```
.uh "Title"
```

which will do a section heading, but will not put a number on the section.

Section Heading Reference

`.sh` *+N T a b c d e f*

Begin numbered section of depth *N*. If *N* is missing, the current depth (maintained in the number register `\n($0)` is used. The values of the individual parts of the section number are maintained in `\n($1` through `\n($6`. There is a `\n(ss` [1v] space before the section. *T* is printed as a section title in font `\n(sf` [8] and size `\n(sp` [10p]. The 'name' of the section may be accessed via `\*($n`. If `\n(si` is non-zero, the base indent is set to `\n(si` times the section depth, and the section title is exdented (see `.ba` in "Miscellaneous Requests"). Also, an additional indent of `\n(so` [0] is added to the section title but not to the body of the section. The font is then set to the paragraph font, so that more information may occur on the line with the section number and title. A `.sh` insures that there is enough room to print the section head plus the beginning of a paragraph, which is about 3 lines total. If you specify *a* through *f*, the section number is set to that number rather than incremented automatically. If any of *a* through *f* are a hyphen that number is not reset. If *T* is a single underscore (_), the section depth

and numbering is reset, but the base indent is not reset and nothing is printed. This is useful to automatically coordinate section numbers with chapter numbers.

`.sx +N` Go to section depth '*N* [−1]', but do not print the number and title, and do not increment the section number at level *N*. This has the effect of starting a new paragraph at level *N*.

`.uh` *T* Unnumbered section heading. The title *T* is printed with the same rules for spacing, font, etc., as for `.sh`.

`.$p` *T B N*

Print section heading. May be redefined to get fancier headings. *T* is the title passed on the `.sh` or `.uh` line; *B* is the section number for this section, and *N* is the depth of this section. These parameters are not always present; in particular, `.sh` passes all three, `.uh` passes only the first, and `.sx` passes three, but the first two are null strings. Be careful if you redefine this macro, as it is quite complex and subtle.

`.$0` *T B N*

Called automatically after every call to `.$p`. It is normally undefined, but may be used to put every section title automatically into the table of contents, or for some similar function. *T* is the section title for the section title just printed, *B* is the section number, and *N* is the section depth.

`.$1 – .$6`

Traps called just before printing that depth section. May be defined to give variable spacing before sections. These macros are called from `.$p`, so if you redefine that macro you may lose this feature.

Parts of the Standard Paper

Some requests help you to format papers. The `.tp` request initializes for a title page. There are no headers or footers on a title page, and unlike other pages, you can space down and leave blank space at the top. For example, source for a typical title page might be:

```
.tp
.sp 2i
.(l C
A BENCHMARK FOR THE NEW SYSTEM
.sp
by
.sp
J. P. Hacker
.)l
.bp
```

The request `.th` sets up the environment of the `nroff` processor to do a thesis. It defines the correct headers, footers, a page number in the upper right-hand corner only, sets the margins correctly, and double spaces.

Use the `.+c` *T* request to start chapters. Each chapter is automatically numbered from one, and a heading is printed at the top of each chapter with the chapter number and the chapter name *T*. For example, to begin a chapter called *Conclusions*, use the request:

```
.+c "CONCLUSIONS"
```

which produces on a new page, the lines

CONCLUSIONS

with appropriate spacing for a thesis. Also, the header is moved to the foot of the page on the first page of a chapter. Although the `.+c` request was not designed to work only with the `.th` request, it is tuned for the format acceptable for a standard PhD thesis.

If the title parameter *T* is omitted from the `.+c` request, the result is a chapter with no heading. You can also use this at the beginning of a paper.

Although papers traditionally have the abstract, table of contents, and so forth at the front, it is more convenient to format and print them last when using `nroff`. This is so that index entries can be collected and then printed for the table of contents. At the end of the paper, give the `.++ P` request, which begins the preliminary part of the paper. After using this request, the `.+c` request will begin a preliminary section of the paper. Most notably, this prints the page number restarted from one in lower case Roman numbers. You may use `.+c` repeatedly to begin different parts of the front material for example, the abstract, the table of contents, acknowledgments, list of illustrations, and so on. You may also use the request `.++ B` to begin the bibliographic section at the end of the paper. For example, the paper might appear as outlined below. (In this figure, comments begin with the sequence `\"`.)

Figure 4-1 *Outline of a Sample Paper*

```
.th                                 \" set for thesis mode
.fo ''DRAFT''                       \" define footer for each page
.tp                                 \" begin title page
.(l C                               \" center a large block
A BENCHMARK FOR THE NEW SYSTEM
.sp
by
.sp
J.P. Hacker
.)l                                 \" end centered part
.+c INTRODUCTION                    \" begin chapter named 'INTRODUCTION'
.(x t                               \" make an entry into index 't'
Introduction
.)x                                 \" end of index entry
text of chapter one
.+c "NEXT CHAPTER"                  \" begin another chapter
.(x t                               \" enter into index 't' again
Next Chapter
.)x
text of chapter two
.+c CONCLUSIONS
.(x t
Conclusions
.)x
text of chapter three
.++ B                               \" begin bibliographic information
.+c BIBLIOGRAPHY                    \" begin another 'chapter'
.(x t
Bibliography
.)x
text of bibliography
.++ P                               \" begin preliminary material
.+c "TABLE OF CONTENTS"
.xp t                               \" print index 't' collected above
.+c PREFACE                         \" begin another preliminary section
text of preface
```

Standard Paper Reference

`.tp` Begin title page. Spacing at the top of the page can occur, and headers and footers are suppressed. Also, the page number is not incremented for this page.

`.th` Set thesis mode. This defines the modes acceptable for a doctoral dissertation. It double spaces, defines the header to be a single page number, and changes the margins to be 1.5 inch on the left and one inch on the top. Use `.++` and `.+c` with it. This macro must be stated before initialization, that is, before the first call of a paragraph macro or `.H`.

`.++` *m H* This request defines the section of the paper you are typing. The section type is defined by *m*: `C` means you are entering the chapter portion of the paper, `A` means you are entering the appendix portion of the paper, `P` means the material following should be the preliminary portion (abstract, table of contents, etc.) of the paper, `AB` means that you are entering the abstract (numbered independently from 1 in Arabic numerals), and `B` means that you are entering the

bibliographic portion at the end of the paper. You can also use the variants RC and RA, which specify renumbering of pages from one at the beginning of each chapter or appendix, respectively. The H parameter defines the new header. If there are any spaces in it, the entire header must be quoted. If you want the header to have the chapter number in it, use the string \\\\n(ch. For example, to number appendixes 'A.1' etc., type .++ RA ''' \\\\n(ch.%'. Precede each section (chapter, appendix, etc.) by the .+c request. When using troff, it is easier to put the front material at the end of the paper, so that the table of contents can be collected and generated; you can then physically move this material to the beginning of the paper.

.+c *T* Begin chapter with title *T*. The chapter number is maintained in \n(ch. This register is incremented every time .+c is called with a parameter. The title and chapter number are printed by .$c. The header is moved to the footer on the first page of each chapter. If *T* is omitted, .$c is not called; this is useful for doing your own 'title page' at the beginning of papers without a title page proper. .$c calls .$C as a hook so that chapter titles can be inserted into a table of contents automatically. The footnote numbering is reset to one.

.$c *T* Print chapter number (from \n(ch) and *T*. You can redefine this macro to your liking. It is defined by default to be acceptable for a standard PhD thesis. This macro calls $C, which can be defined to make index entries, or whatever.

.$C *K N T*

This macro is called by .$c. It is normally undefined, but can be used to automatically insert index entries, or whatever. *K* is a keyword, either 'Chapter' or 'Appendix' (depending on the .++ mode); *N* is the chapter or appendix number, and *T* is the chapter or appendix title.

.ac *A N* This macro (short for .acm) sets up the nroff environment for photo-ready papers as used by the Association for Computing Machinery (ACM). This format is 25% larger, and has no headers or footers. The author's name *A* is printed at the bottom of the page, but off the part which will be printed in the conference proceedings, together with the current page number and the total number of pages *N*. Additionally, this macro loads the file */usr/lib/me/acm.me*, which may later be augmented with other macros for printing papers for ACM conferences. Note that this macro will not work correctly in troff, since it sets the page length wider than the physical width of the phototypesetter roll.

Two-Column Output — .2c

You can get two-column output automatically by using the request .2c. This produces everything after it in two-column form. The request .bc will start a new column; it differs from .bp in that .bp may leave a totally blank column when it starts a new page. To revert to single-column output, use .1c.

Column Output Reference

.2c +*S N*
: Enter two-column mode. The column separation is set to +*S* [4n, 0.5i in ACM mode] (saved in \n($s). The column width, calculated to fill the single-column line length with both columns, is stored in \n($l. The current column is in \n($c. You can test register \n($m [1] to see if you are in single-column or double-column mode. Actually, the request enters *N*-column [2] output.

.1c
: Revert to single-column mode.

.bc
: Begin column. This is like .bp except that it begins a new column on a new page only if necessary, rather than forcing a whole new page if there is another column left on the current page.

Defining Macros — .de

A *macro* is a collection of requests and text which you may invoke with a simple request. Macros definitions begin with the line .de *xx* where *xx* is the name of the macro to be defined, and end with a line consisting of only two dots. After defining the macro, invoking it with the line .*xx* is the same as invoking all the other macros. For example, to define a macro that spaces vertically three lines and then centers the next input line, type:

```
.de SS
.sp 3
.ce
..
```

and use it by typing:

```
.SS
Title Line
(beginning of text)
```

Macro names may be one or two characters. In order to avoid conflicts with command names in -me, always use upper case letters as names. Avoid the names TS, TH, TE, EQ, and EN.

Annotations Inside Keeps

Sometimes you may want to put a footnote or index entry inside a keep. For example, if you want to maintain a 'list of figures', you will want to use something like:

```
.(z
.(c
Text of figure
.)c
.ce
Figure 5.
\!.(x f
\!Figure 5
\!.)x
.)z
```

which will give you a figure with a label and an entry in the index 'f', presumably a list of figures index. Because the index entry is read and interpreted when the keep is read, and not when it is printed, you have is to use the magic string `\!` at the beginning of all the lines dealing with the index. Otherwise, the page number in the index is likely to be wrong. This defers index processing until the figure is generated, and guarantees that the page number in the index is correct. The same comments apply to blocks with `.(b` and `.)b`.

4.6. Using `troff` for Phototypesetting

You can prepare documents for either displaying on a workstation or for phototypesetting using the `troff` formatting program.

Fonts

A *font* is a style of type. There are three fonts that are available simultaneously, Times Roman, Times Italic, and Times Bold, plus the special math font for use with the `eqn` and `neqn` mathematical equation processors. The normal font is Roman. Text which would be underlined in `nroff` with the `.ul` request is set in italics in `troff`.

There are ways of switching between fonts. The requests `.r`, `.i`, and `.b` switch to Roman, italic, and bold fonts respectively. You can set a single word in one of these fonts by typing, for example:

```
.i word
```

which will set *word* in italics but does not affect the surrounding text. In `nroff`, italic and bold text is underlined.

Notice that if you are setting more than one word in a different font, you must surround that word with double quote marks (") so it will appear to the `nroff` processor as a single word. The quote marks will not appear in the formatted text. If you do want a quote mark to appear, quote the entire string even if a single word, and use *two* quote marks where you want one to appear. For example, if you want to produce the text:

"Master Control"

in italics, you must type:

```
.i """Master Control\|"""
```

The \| produces a narrow space so that the 'l' does not overlap the quote sign in `troff`.

There are also several *pseudo-fonts* available. For example, the input:

```
.u underlined
```

generates

underlined

and

```
.bx "words in a box"
```

produces

words in a box

You can also get bold italics with

```
.bi "bold italics"
```

Notice that pseudo font requests set only the single parameter in the pseudo font; ordinary font requests will begin setting all text in the special font if you do not provide a parameter. No more than one word should appear with these three font requests in the middle of lines. This is because of the way `troff` justifies text. For example, if you were to give the requests:

```
.bi "some bold italics"
```

and

```
.bx "words in a box"
```

in the middle of a line, `troff` would overwrite the first and the box lines on the second would be poorly drawn.

The second parameter of all font requests is set in the original font. For example, the font request:

```
.b bold face
```

generates 'bold' in bold font, but sets 'face' in the font of the surrounding text, resulting in:

boldface

To set the two words 'bold' and 'face' both in **bold face**, type:

```
.b "bold face"
```

You can mix fonts in a word by using the special sequence `\c` at the end of a line to indicate 'continue text processing'; you can join input lines together without a space between them. For example, the input:

```
.u under \c
.i italics
```

generates under*italics* , but if you type:

```
.u under
.i italics
```

the result is under *italics* as two words.

Point Sizes — `.sz`

The phototypesetter supports different sizes of type, measured in points. The default point size is 10 points for most text and eight points for footnotes. To change the point size, type:

```
.sz +N
```

where *N* is the size wanted in points. The 'vertical spacing,' that is, the distance between the bottom of most letters (the *baseline*) and the adjacent line is set to be proportional to the type size.

Note: Changing point sizes on the phototypesetter is a slow mechanical operation. Consider size changes carefully.

Fonts and Sizes Reference

`.sz +P` The point size is set to *P* [10p], and the line spacing is set proportionally. The ratio of line spacing to point size is stored in **\n($r**. The ratio used internally by displays and annotations is stored in **\n($R**, although `.sz` does not use this.

`.r` *W X* Set *W* in roman font, appending *X* in the previous font. To append different font requests, use '*X* = `\c`. If no parameters, change to roman font.

`.i` *W X* Set *W* in italics, appending *X* in the previous font. If no parameters, change to italic font. Underlines in `nroff`.

`.b` *W X* Set *W* in bold font and append *X* in the previous font. If no parameters, switch to bold font. Underlines in `nroff`.

`.rb` *W X* Set *W* in bold font and append *X* in the previous font. If no parameters, switch to bold font. `.rb` differs from `.b` in that `.rb` does not underline in `nroff`.

`.u` *W X* Underline *W* and append *X*. This is a true underlining, as opposed to the `.ul` request, which changes to 'underline font' (usually italics in

troff). It won't work right if *W* is spread or broken, which includes being hyphenated, so in other words, it is only safe in nofill mode.

`.q` *W X* Quote *W* and append *X*. In `nroff` this just surrounds *W* with double quote marks (""), but in `troff` uses directed quotes.

`.bi` *W X* Set *W* in bold italics and append *X*. Actually, sets *W* in italic and overstrikes once. Underlines in `nroff`. It won't work right if *W* is spread or broken, which includes being hyphenated, so it is only safe in nofill mode.

`.bx` *W X* Sets *W* in a box, with *X* appended. Underlines in `nroff`. It won't work right if *W* is spread or broken, which includes being hyphenated, so it is only safe in nofill mode.

Quotes — `\*(lq` and `\*(rq`

It looks better to use pairs of grave and acute accents to generate double quotes, rather than the double quote character (") on a phototypesetter. For example, compare "quote" to "quote". In order to make quotes compatible between the typesetter and the workstation or a terminal, use the sequences `\*(lq` and `\*(rq` to stand for the left and right quote respectively. These both appear as " on most terminals, but are typeset as `` and '' respectively. For example, use:

```
\*(lqSome things aren't true
even if they did happen.\*(rq
```

to generate the result:

"Some things aren't true even if they did happen."

As a shorthand, the special font request:

```
.q "quoted text"
```

which generates "quoted text". Notice that you must surround the material to be quoted with double quote marks if it is more than one word.

4.7. Adjusting Macro Parameters

You may adjust a number of macro parameters. You may set fonts to a font number only. In `nroff` font 8 is underlined, and is set in bold font in `troff` (although font 3, bold in `troff`, is not underlined in `nroff`). Font 0 is no font change; the font of the surrounding text is used instead. Notice that fonts 0 and 8 are *pseudo-fonts*; that is, they are simulated by the macros. This means that although it is legal to set a font register to zero or eight, it is not legal to use the escape character form, such as:

```
\f8
```

All distances are in basic units, so it is nearly always necessary to use a scaling factor. For example, the request to set the paragraph indent to eight one-en spaces is:

```
.nr pi 8n
```

and not

```
.nr pi 8
```

which would set the paragraph indent to eight basic units, or about 0.02 inch.

You may use registers and strings of the form `$` *x* in expressions but you should not change them. Macros of the form `$` *x* perform some function as described and may be redefined to change this function. This may be a sensitive operation; look at the body of the original macro before changing it.

On daisy wheel printers in twelve-pitch, you can use the `-rx1` flag to make lines default to one-eighth inch, which is the normal spacing for a newline in twelve-pitch. This is normally too small for easy readability, so the default is to space one-sixth inch.

4.8. `roff` Support

`.ix` +*N*	Indent, no break. Equivalent to ''*N*'.
`.bl` *N*	Leave *N* contiguous white spaces, on the next page if not enough room on this page. Equivalent to a `.sp` *N* inside a block.
`.pa` +*N*	Equivalent to `.bp`.
`.ro`	Set page number in Roman numerals. Equivalent to `.af % i`.
`.ar`	Set page number in Arabic. Equivalent to `.af % 1`.
`.nl`	Number lines in margin from one on each page.
`.n2` *N*	Number lines from *N*, stop if N = 0.
`.sk`	Leave the next output page blank, except for headers and footers. Use this to leave space for a full-page diagram which is produced externally and pasted in later. To get a partial-page paste-in display, say `.sv` *N*, where *N* is the amount of space to leave; this space will be generated immediately if there is room, and will otherwise be generated at the top of the next page. However, be warned: if *N* is greater than the amount of available space on an *empty* page, no space will be reserved.

4.9. Preprocessor Support

`.EQ` *m T*	Begin equation. The equation is centered if *m* is `C` or omitted, indented `\n(bi` [4n] if *m* is `I`, and left-justified if *m* is `L`. *T* is a title printed on the right margin next to the equation. See the "Typesetting Mathematics with `eqn`" chapter in this manual for more about equation formatting.
`.EN` *c*	End equation. If *c* is `C`, the equation must be continued by immediately following with another `.EQ`, the text of which can be centered along with this one. Otherwise, the equation is printed, always on one page, with `\n(es` [0.5v in `troff`, 1v in `nroff`] space above and below it.
`.TS` *h*	Table start. Tables are single-spaced and kept on one page, if possible. If you have a large table that will not fit on one page, use *h* = `H` and follow the header part to be printed on every page of the table with a `.TH`. See the "Formatting Tables with `tbl`" chapter in this manual for more information on laying out tables.
`.TH`	With `.TS H`, ends the header portion of the table.
`.TE`	Table end. Note that this table does not float, in fact, it is not even guaranteed to stay on one page if you use requests such as `.sp` intermixed with the text of the table. If you want it to float (or if you use requests inside the table), surround the entire table (including the `.TS` and `.TE` requests) with `.(z` and `.)z`.

4.10. Predefined Strings

`\**`	Footnote number, actually `\*[\n($f\*]`. This macro is incremented after each call to `.)f`.
`\*#`	Delayed text number. Actually `[\n($d]`.
`\*[`	Superscript. This string gives upward movement and a change to a smaller point size if possible, otherwise it gives the left bracket character ([). Extra space is left above the line to allow room for the superscript. For example, to produce a superscript you can type `x\*[2\*]`, which will produce x^2.
`\*]`	Unsuperscript. Inverse of `\*[`.
`\*<`	Subscript. Defaults to < if half-carriage motion not possible. Extra space is left below the line to allow for the subscript.
`\*>`	Inverse of `\*<`.
`\*(dw`	The day of the week, as a word.
`\*(mo`	The month, as a word.
`\*(td`	Today's date, directly printable. The date is of the form September 16, 1983. Other forms of the date can be used by using `\n(dy` (the day of the month; for example, 16), `\*(mo` (as noted above) or `\n(mo` (the same, but as an ordinal number; for example, September is 9), and `\n(yr` (the last two digits of the current year).
`\*(lq`	Left quote marks; double quote in `nroff`.
`\*(rq`	Right quote marks; double quote in `nroff`.
`\*-`	An em-dash in `troff`; two hyphens in `nroff`.

4.11. Miscellaneous Requests

`.re`	Reset tabs. Set to every 0.5i in `troff` and every 0.8i in `nroff`.
`.ba +`*N*	Set the base indent to +N [0] (saved in `\n($i`). All paragraphs, sections, and displays come out indented by this amount. Titles and footnotes are unaffected. The `.H` request performs a `.ba` request if `\n(si` [0] is not zero, and sets the base indent to `\n(si*\n($0`.
`.xl +`*N*	Set the line length to *N* [6.0i]. This differs from `.ll` because it only affects the current environment.
`.ll +`*N*	Set line length in all environments to *N* [6.0i]. Do not use this after output has begun, and particularly not in two-column output. The current line length is stored in `\n($l`.
`.hl`	Draws a horizontal line the length of the page. This is useful inside floating keeps to differentiate between the text and the figure.
`.lo`	This macro loads another set of macros in `/usr/lib/me/local.me`, which is a set of locally-defined macros. These macros should all be of the form `.*` *X*, where *X* is any letter (upper or lower case) or digit.

4.12. Special Characters and Diacritical Marks — .sc

There are a number of special characters and diacritical marks, such as accents, available with -me. To use these characters, you must call the macro .sc to define the characters before using them.

.sc Define special characters and diacritical marks. You must state this macro before initialization.

The special characters available are listed below.

Table 4-1 *Special Characters and Diacritical Marks*

Name	Usage	Example
Acute accent	*'	a*' á
Grave accent	*`	e*` è
Umlaut	*:	u*: ü
Tilde	*~	n*~ ñ
Caret	*^/e*^ ê	
Cedilla	*,/c*, ç	
Czech	*v/e*v/e	
Circle	*o	A*o Å

4.13. -me Request Summary

Table 4-2 -me *Request Summary*

Request	*Initial Value*	*Cause Break*	*Explanation*
.(c	–	yes	Begin centered block.
.(d	–	no	Begin delayed text.
.(f	–	no	Begin footnote.
.(l	–	yes	Begin list.
.(q	–	yes	Begin major quote.
.(x *x*	–	no	Begin indexed item in index *x*.
.(z	–	no	Begin floating keep.
.)c	–	yes	End centered block.
.)d	–	yes	End delayed text.
.)f	–	yes	End footnote.
.)l	–	yes	End list.
.)q	–	yes	End major quote.
.)x	–	yes	End index item.
.)z	–	yes	End floating keep.
.++ *m H*	–	no	Define paper section. *m* defines the part of the paper and can be C (chapter), A (appendix), P (preliminary, for example, abstract, table of contents, etc.), B (bibliography), RC (chapters renumbered from page one each chapter), or RA (appendix renumbered from page one).

Table 4-2 -me *Request Summary— Continued*

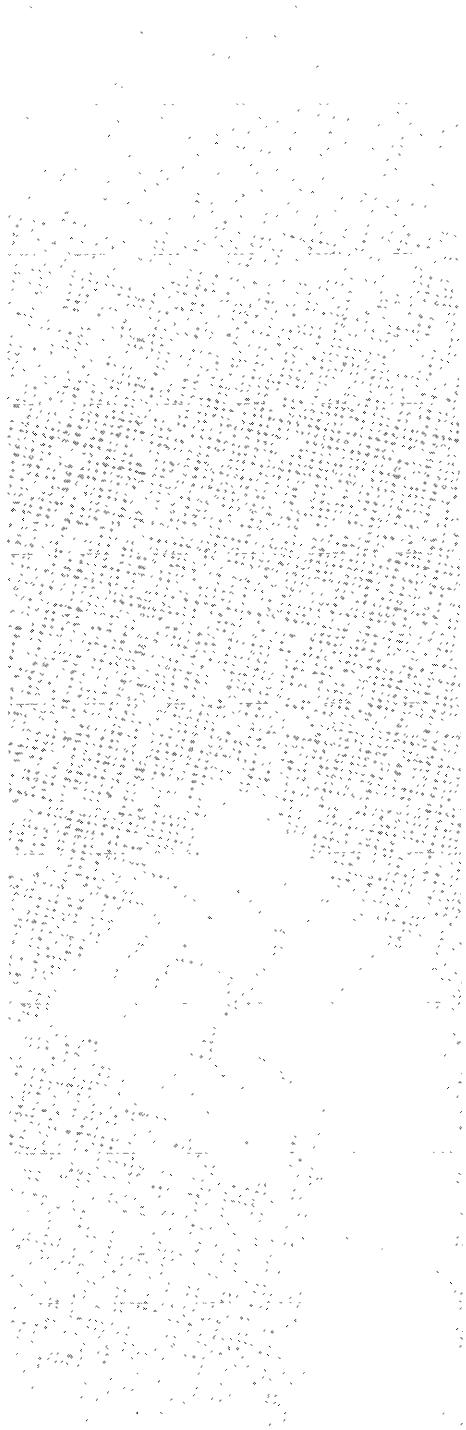
Request	*Initial Value*	*Cause Break*	*Explanation*
.+c *T*	–	yes	Begin chapter (or appendix, etc., as set by .++). *T* is the chapter title.
.1c	1	yes	One-column format on a new page.
.2c	1	yes	Two-column format.
.EN	–	yes	Space after equation produced by eqn or neqn.
.EQ *x y*	–	yes	Precede equation; break out and add space. Equation number is *y*. The optional argument *x* may be I to indent equation (default), L to left-adjust the equation, orC to center the equation.
.TE	–	yes	End table.
.TH	–	yes	End heading section of table.
.TS *x*	–	yes	Begin table; if *x* is H, table has repeated heading.
.ac *A N*	–	no	Set up for ACM-style output. *A* is the Author's name(s), *N* is the total number of pages. Must be given before the first initialization.
.b *x*	no	yes	Print *x* in boldface; if no argument switch to boldface.
.ba +*n*	0	yes	Augments the base indent by *n*. This indent is used to set the indent on regular text (like paragraphs).
.bc	no	yes	Begin new column.
.bi *x*	no	no	Print *x* in bold italics (nofill only).
.bx *x*	no	no	Print *x* in a box (nofill only).
.ef *'x'y'z'*	''''	no	Set even footer to *x y z*.
.eh *'x'y'z'*	''''	no	Set even header to *x y z*.
.fo *'x'y'z'*	''''	no	Set footer to *x y z*.
.he *'x'y'z'*	''''	no	Set header to *x y z*.
.hl	–	yes	Draw a horizontal line.
.hx	–	no	Suppress headers and footers on next page.
.i *x*	no	no	Italicize *x*; if *x* is missing, italic text follows.
.ip *x y*	no	yes	Start indented paragraph, with hanging tag *x*. Indentation is *y* ens (default 5).
.lp	yes	yes	Start left-block paragraph.
.lo	–	no	Read in a file of local macros of the form .*x. Must be given before initialization.
.np	1	yes	Start numbered paragraph.
.of *'x'y'z'*	''''	no	Set odd footer to *x y z*.
.oh *'x'y'z'*	''''	no	Set odd header to *x y z*.
.pd	–	yes	Print delayed text.
.pp	no	yes	Begin paragraph. First line indented.
.r	yes	no	Roman text follows.
.re	–	no	Reset tabs to default values.
.sc	–	no	Read in a file of special characters and diacritical marks. Must be given before initialization.

Table 4-2 -me *Request Summary— Continued*

Request	*Initial Value*	*Cause Break*	*Explanation*
`.sh` *n x*	–	yes	Section head follows, font automatically bold. *n* is level of section, *x* is title of section.
`.sk`	no	no	Leave the next page blank. Only one page is remembered ahead.
`.sz` + *n*	10p	no	Increase the point size by *n* points.
`.th`	no	no	Produce the paper in thesis format. Must be given before initialization.
`.tp`	no	yes	Begin title page.
`.u` *x*	–	no	Underline argument (even in `troff`) (nofill only).
`.uh`	–	yes	Like '.sh' but unnumbered.
`.xp` *x*	–	no	Print index *x*.

5

refer — A Bibliography System



5

refer — A Bibliography System

5.1. Introduction

refer is a bibliography system that supports data entry, indexing, retrieval, sorting, runoff, convenient citations, and footnote or endnote numbering. You can enter new bibliographic data into the database, index the selected data, and retrieve bibliographic references from the database. This document assumes you know how to use a Unix editor, and that you are familiar with the nroff and troff text formatters.

The refer program is a preprocessor for nroff and troff, and works like like eqn and tbl. refer is used for literature citations, rather than for equations and tables. Given incomplete but sufficiently precise citations, refer finds references in a bibliographic database. The complete references are formatted as footnotes, numbered, and placed either at the bottom of the page, or at the end of a chapter.

A number of related programs make refer easier to use. The addbib program is for creating and extending the bibliographic database; sortbib sorts the bibliography by author and date, or other selected criteria; and roffbib runs off the entire database, formatting it not as footnotes, but as a bibliography or annotated bibliography.

Once a full bibliography has been created, access time can be improved by making an index to the references with indxbib. Then, the lookbib program can be used to quickly retrieve individual citations or groups of citations. Creating this inverted index will speed up refer, and lookbib will allow you to verify that a citation is sufficiently precise to deliver just one reference.

5.2. Features

Taken together, the refer programs constitute a database system for use with variable-length information. To distinguish various types of bibliographic material, the system uses *labels* composed of upper case letters, preceded by a percent sign and followed by a space. For example, one document might be given this entry:

```
%A Joel Kies
%T Document Formatting on Unix Using the -ms Macros
%I Computing Services
%C Berkeley
%D 1980
```

Each line is called a *field*, and lines grouped together are called a *record*; records are separated from each other by a blank line. Bibliographic information follows the labels. This field contains *data* to be used by the `refer` system. The order of fields is not important, except that authors should be entered in the same order as they are listed on the document. Fields can be as long as necessary, and may even be continued on the following line(s).

The labels are meaningful to `nroff` and `troff` macros, and, with a few exceptions, the `refer` program itself does not pay attention to the labels. This implies that you can change the label codes, if you also change the macros used by `nroff` and `troff`. The macro package takes care of details like proper ordering, underlining the book title or journal name, and quoting the article's title. Here are the labels used by `refer`, with an indication of what they represent:

%H	Header commentary, printed before reference
%A	Author's name
%Q	Corporate or foreign author (unreversed)
%T	Title of article or book
%S	Series title
%J	Journal containing article
%B	Book containing article
%R	Report, paper, or thesis (for unpublished material)
%V	Volume
%N	Number within volume
%E	Editor of book containing article
%P	Page number(s)
%I	Issuer (publisher)
%C	City where published
%D	Date of publication
%O	Other commentary, printed at end of reference
%K	Keywords used to locate reference
%L	Label used by –k option of `refer`
%X	Abstract (used by `roffbib`, not by `refer`)

Only relevant fields (lines) should be supplied. Except for %A, the author field, each field should be given only once. In the case of multiple authors, the senior author should be entered first. Your entry in such a case, might look like this:

```
%A Brian W. Kernighan
%A P. J. Plauger
%T Software Tools in Pascal
%I Addison-Wesley
%C Reading, Massachusetts
%D 1981
```

The %Q is for organizational authors, or authors with Japanese or Arabic names, in which cases there is no clear last name. Books should be labeled with the %T, not with the %B, which is reserved for books containing articles. The %J and %B fields should never appear together, although if they do, the %J will override the %B. If there is no author, just an editor, it is best to type the editor in the %A

field, as in this example:

```
%A Bertrand Bronson, ed.
```

The %E field is used for the editor of a book (%B) containing an article, which has its own author. For unpublished material such as theses, use the %R field; the title in the %T field will be quoted, but the contents of the %R field will not be underlined. Unlike other fields, %H, %O, and %X should contain their own punctuation. Here is an example:

```
%A Mike E. Lesk
%T Some Applications of Inverted Indexes on the Unix System
%B Unix Programmer's Manual
%I Bell Laboratories
%C Murray Hill, NJ
%D 1978
%V 2a
%K refer mkey inv hunt
%X Difficult to read paper that dwells on indexing strategies,
giving little practical advice about using \fBrefer\fP.
```

Note that the author's name is given in normal order, without inverting the surname; inversion is done automatically, except when %Q is used instead of %A. We use %X rather than %O for the commentary because we do not want the comment printed every time the reference is used. The %O and %H fields are printed by both `refer` and `roffbib`; the %X field is printed only by `roffbib`, as a detached annotation paragraph.

5.3. Data Entry with `addbib`

The `addbib` program is for creating and extending bibliographic databases. You must give it the filename of your bibliography:

```
hostname% addbib database
```

Every time you enter `addbib`, it asks if you want instructions. To get them, type **y**; to skip them, type RETURN. `addbib` prompts for various fields, reads from the keyboard, and writes records containing the `refer` codes to the database. After finishing a field entry, you should end it by typing RETURN. If a field is too long to fit on a line, type a backslash (\) at the end of the line, and you will be able to continue on the following line. Note: the backslash works in this capacity only inside `addbib`.

A field will not be written to the database if nothing is entered into it. Typing a minus sign as the first character of any field will cause `addbib` to back up one field at a time. Backing up is the best way to add multiple authors, and it really helps if you forget to add something important. Fields not contained in the prompting skeleton may be entered by typing a backslash as the last character before RETURN. The following line will be sent verbatim to the database and `addbib` will resume with the next field. This is identical to the procedure for dealing with long fields, but with new fields, don't forget the % key-letter.

Finally, you will be asked for an abstract (or annotation), which will be preserved as the %X field. Type in as many lines as you need, and end with a control-D (hold down the CTRL button, then press the "d" key). This prompting for an abstract can be suppressed with the -a command line option.

After one bibliographic record has been completed, `addbib` will ask if you want to continue. If you do, type RETURN; to quit, type **`q`** or **`n`** (quit or no). It is also possible to use one of the system editors to correct mistakes made while entering data. After the `Continue?` prompt, type any of the following: **`edit`**, **`ex`**, **`vi`**, or **`ed`** — you will be placed inside the corresponding editor, and returned to `addbib` afterwards, from where you can either quit or add more data.

If the prompts normally supplied by `addbib` are not enough, are in the wrong order, or are too numerous, you can redefine the skeleton by constructing a promptfile. Create some file, to be named after the -p command line option. Place the prompts you want on the left side, followed by a single TAB (control-I), then the `refer` code that is to appear in the bibliographic database. `addbib` will send the left side to the screen, and the right side, along with data entered, to the database.

5.4. Printing the Bibliography

`sortbib` is for sorting the bibliography by author (%A) and date (%D), or by data in other fields. Sortbib is quite useful for producing bibliographies and annotated bibliographies, which are seldom entered in strict alphabetical order.

Sortbib takes as arguments the names of up to 16 bibliography files, and sends the sorted records to standard output (the terminal screen), which may be redirected through a pipe or into a file.

The -s*KEYS* flag to `sortbib` will sort by fields whose key-letters are in the *KEYS* string, rather than merely by author and date. Key-letters in *KEYS* may be followed by a + to indicate that all such fields are to be used. The default is to sort by senior author and date (printing the senior author last name first), but `-sA+D` will sort by all authors and then date, and `-sATD` will sort on senior author, then title, and then date.

`roffbib` is for running off the (probably sorted) bibliography. It can handle annotated bibliographies — annotations are entered in the %X (abstract) field. `roffbib` is a shell script that calls `refer -B` and `nroff -mbib`. It uses the macro definitions that reside in `/usr/lib/tmac/tmac.bib`, which you can redefine if you know `nroff` and `troff`. Note that `refer` will print the %H and %O commentaries, but will ignore abstracts in the %X field; `roffbib` will print both fields, unless annotations are suppressed with the -x option.

The following command sequence will lineprint the entire bibliography, organized alphabetically by author and date:

```
hostname% sortbib database | roffbib | lpr
```

This is a good way to proofread the bibliography, or to produce a stand-alone bibliography at the end of a paper. Incidentally, `roffbib` accepts all flags used

with nroff. For example:

```
hostname% sortbib database | roffbib -Txerox -s1
```

will make accent marks work on a Xerox printer, and stop at the bottom of every page for changing paper. The -n and -o flags may also be quite useful, to start page numbering at a selected point, or to produce only specific pages.

roffbib understands four command-line number registers: N, V, L, and O. These are something like the two-letter number registers in **-ms**. The -rN1 argument will number references beginning at one (1); use another number to start somewhere besides one. The -rV2 flag will double-space the entire bibliography, while -rV1 will double-space the references, but single-space the annotation paragraphs. Finally, specifying -rL6i changes the line length from 6.5 inches to 6 inches, and saying -rO1i sets the page offset to one inch, instead of zero. (That's a capital O after -r, not a zero.)

5.5. Citing Papers with refer

The refer program normally copies input to output, except when it encounters an item of the form:

```
.[
partial citation
.]
```

The partial citation may be just an author's name and a date, or perhaps a title and a keyword, or maybe just a document number. refer looks up the citation in the bibliographic database, and transforms it into a full, properly-formatted reference. If the partial citation does not correctly identify a single work (either finding nothing, or more than one reference), a diagnostic message is given. If nothing is found, it will say "No such paper." If more than one reference is found, it will say "Too many hits." Other diagnostic messages can be quite cryptic; if you are in doubt, use checknr to verify that all your .[s have matching .] s.

When everything goes well, the reference will be brought in from the database, numbered, and placed at the bottom of the page. This citation, for example, was produced by:

```
This citation,
.[
lesk inverted indexes
.]
for example, was produced by
```

The .[and .] markers, in essence, replace the .FS and .FE of the **-ms** macros, and also provide a numbering mechanism. Footnote numbers will be bracketed

[1] Mike E. Lesk, "Some Applications of Inverted Indexes on the Unix System," in *Unix Programmer's Manual*, Bell Laboratories, Murray Hill, NJ, 1978.

on the lineprinter, but superscripted on daisy-wheel terminals and in `troff`. In the reference itself, articles will be quoted, and books and journals will be underlined in `nroff`, and italicized in `troff`.

Sometimes you need to cite a specific page number along with more general bibliographic material. You may have, for instance, a single document that you refer to several times, each time giving a different page citation. This is how you could get "p. 10" in the reference:

```
.[
kies document formatting
%P 10
.]
```

The first line, a partial citation, will find the reference in your bibliography. The second line will insert the page number into the final citation. Ranges of pages may be specified as "%P 56-78".

When the time comes to run off a paper, you will need to have two files: the bibliographic database, and the paper to format. Use a command line something like one of these:

```
hostname% refer -p database paper | nroff -ms
hostname% refer -p database paper | tbl | nroff -ms
hostname% refer -p database paper | tbl | neqn | nroff -ms
```

If other preprocessors are used, `refer` should precede `tbl`, which must in turn precede `eqn`, or `neqn`. The `-p` option specifies a "private" database, which most bibliographies are.

5.6. `refer` Command Line Options

Many people like to place references at the end of a chapter, rather than at the bottom of the page. The `-e` option will accumulate references until a macro sequence of the form

```
.[
$LIST$
.]
```

is encountered (or until the end of file). `refer` will then write out all references collected up to that point, collapsing identical references. Warning: there is a limit (currently 200) on the number of references that can be accumulated at one time.

It is also possible to sort references that appear at the end of text. The `-s`*KEYS* flag will sort references by fields whose key-letters are in the *KEYS* string, and permute reference numbers in the text accordingly. It is unnecessary to use `-e` with the `-s`*KEYS* flag, since `-s` implies `-e`. See the section "Printing the Bibliography" for additional features of the `-s`*KEYS* flag.

`refer` can also make citations in what is known as the Social or Natural Sciences format. Instead of numbering references, the `-l` (letter ell) flag makes

labels from the senior author's last name and the year of publication. For example, a reference to the paper on Inverted Indexes cited above might appear as [Lesk1978a]. It is possible to control the number of characters in the last name, and the number of digits in the date. For instance, the command line argument `-l6,2` might produce a reference such as [Kernig78c].

Some bibliography standards shun both footnote numbers and labels composed of author and date, requiring some keyword to identify the reference. The `-k` flag indicates that, instead of numbering references, key labels specified on the %L line should be used to mark references.

The `-n` flag means to not search the default reference file, located in `/usr/dict/papers/Rv7man`. Using this flag may make `refer` marginally faster. The `-a`*n* flag will reverse the first *n* author names, printing Jones, J. A. instead of J. A. Jones. Often `-a1` is enough; this will reverse the first and last names of only the senior author. In some versions of `refer` there is also the `-f` flag to set the footnote number to some predetermined value; for example, `-f23` would start numbering with footnote 23.

5.7. Making an Index

Once your database is large and relatively stable, it is a good idea to make an index to it, so that references can be found quickly and efficiently. The `indxbib` program makes an inverted index to the bibliographic database (this program is called `pubindex` in the Bell Labs manual). An inverted index could be compared to the thumb cuts of a dictionary — instead of going all the way through your bibliography, programs can move to the exact location where a citation is found.

`indxbib` itself takes a while to run, and you will need sufficient disk space to store the indexes. But once it has been run, access time will improve dramatically. Furthermore, large databases of several million characters can be indexed with no problem. The program is exceedingly simple to use:

```
hostname% indxbib database
```

Be aware that changing your database will require that you run `indxbib` over again. If you don't, you may fail to find a reference that really is in the database.

Once you have built an inverted index, you can use `lookbib` to find references in the database. `lookbib` cannot be used until you have run `indxbib`. When editing a paper, `lookbib` is very useful to make sure that a citation can be found as specified. It takes one argument, the name of the bibliography, and then reads partial citations from the terminal, returning references that match, or nothing if none match. Its prompt is the greater-than sign.

```
hostname% lookbib database
Instructions? n
> lesk inverted indexes
%A Mike E. Lesk
%T Some Applications of Inverted Indexes on the Unix System
%J Unix Programmer's Manual
%I Bell Laboratories
%C Murray Hill, NJ
%D 1978
%V 2a
%X Difficult to read paper that dwells on indexing strategies,
giving little practical advice about using \fLrefer\fP.
>
```

If more than one reference comes back, you will have to give a more precise citation for `refer`. Experiment until you find something that works; remember that it is harmless to overspecify.

To get out of the `lookbib` program, type a CTRL-D alone on a line; `lookbib` then exits with an "EOT" message.

`lookbib` can also be used to extract groups of related citations. For example, to find all the papers by Brian Kernighan in the system database, and send the output to a file, type:

```
hostname% lookbib /usr/dict/papers/Ind > kern.refs
Instructions ? n
> kernighan
> CTRL-D
EOT
hostname% cat kern.refs
```

Your file, "kern.refs", will be full of references. A similar procedure can be used to pull out all papers of some date, all papers from a given journal, all papers containing a certain group of keywords, etc.

5.8. `refer` Bugs and Some Solutions

Blanks at Ends of Lines

The `refer` program will mess up if there are blanks at the end of lines, especially the %A author line. `addbib` carefully removes trailing blanks, but they may creep in again during editing. Use an `ex` editor command —

```
g/ *$/s///
```

— or similar method to remove trailing blanks from your bibliography.

Interpolated Strings

Having bibliographic fields passed through as string definitions implies that interpolated strings (such as accent marks) must have two backslashes, so they can pass through copy mode intact. For instance, the word "téléphone" would have to be represented:

```
te\\*'le\\*'phone
```

in order to come out correctly. In the %X field, by contrast, you will have to use single backslashes instead. This is because the %X field is not passed through as a string, but as the body of a paragraph macro.

Interpreting Foreign Surnames

Another problem arises from authors with foreign names. When a name like "Valéry Giscard d'Estaing" is turned around by the –a option of `refer`, it will appear as "d'Estaing, Valéry Giscard," rather than as "Giscard d'Estaing, Valéry." To prevent this, enter names as follows:

```
%A Vale\\*'ry Giscard\0d'Estaing
%A Alexander Csoma\0de\0Ko\\*:ro\\*:s
```

(The second is the name of a famous Hungarian linguist.) The backslash-zero is an `nroff` and `troff` request meaning to insert a digit-width space. Because the second argument to the %A field contains no blank spaces to confuse the `refer` program, `refer` will treat the second field as a single word. This protects against faulty name reversal, and also against mis-sorting.

Footnote Numbers

Footnote numbers are placed at the end of the line before the .[macro. This line should be a line of text, not a macro. As an example, if the line before the `.[` is a `.R` macro, then the `.R` will eat the footnote number. (The `.R` is an **-ms** request meaning change to Roman font.) In cases where the font needs changing, it is necessary to use the following method immediately before the citation:

```
Aho \fIet al.\fP
.[
awk aho kernighan weinberger
.]
```

Now the reference will be to Aho *et al.*[2] The `\fI` changes to italics, and the `\fR` changes back to Roman font. Both these requests are `nroff` and `troff` requests, not part of **-ms**. If and when a footnote number is added after this sequence, it will indeed appear in the output.

[2] Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger, *Awk — A Pattern Scanning and Text Processing Language*, Bell Laboratories, Murray Hill, NJ.

5.9. Internal Details of `refer`

You have already read everything you need to know in order to use the `refer` bibliography system. The remaining sections are provided only for extra information, and in case you need to change the way `refer` works.

The output of `refer` is a stream of string definitions, one for each field in a reference. To create string names, percent signs are simply changed to an open bracket, and an [F string is added, containing the footnote number. The %X, %Y and %Z fields are ignored; however, the `annobib` program changes the %X to an `.AP` (annotation paragraph) macro. The Lesk citation used above yields this intermediate output:

```
.ds [F 1
.]-
.ds [A Mike E. Lesk
.ds [T Some Applications of Inverted Indexes on the Unix System
.ds [J Unix Programmer's Manual
.ds [I Bell Laboratories
.ds [C Murray Hill, NJ
.ds [D 1978
.ds [V 2a
.nr [T 0
.nr [A 0
.nr [O 0
.][ 1 journal-article
```

These string definitions are sent to `nroff`, which can use the **-ms** macros defined in `/usr/lib/mx/ms.xref` to take care of formatting things properly. The initializing macro `.]-` precedes the string definitions, and the labeled macro `.][` follows. These are changed from the input `.[` and `.]` so that running a file twice through `refer` is harmless.

The `.][` macro, used to print the reference, is given a type-number argument, which is a numeric label indicating the type of reference involved. Here is a list of the various kinds of references:

Field	Value	Kind of Reference
%J	1	Journal Article
%B	3	Article in Book
%G	4	Report, Government Report
%I	2	Book
%M	5	Bell Labs Memorandum (undefined)
none	0	Other

The order listed above is indicative of the precedence of the various fields. In other words, a reference that has both the %J and %B fields will be classified as a journal article. If none of the fields listed is present, then the reference will be classified as "other."

The footnote number is flagged in the text with the following sequence, where *number* is the footnote number:

```
\*([.number\*(.]
```

The *([. and *(.] stand for bracketing or superscripting. In `nroff` with low-resolution devices such as the lpr and a crt, footnote numbers will be bracketed. In `troff`, or on daisy-wheel printers, footnote numbers will be superscripted. Punctuation normally comes before the reference number; this can be changed by using the `-P` (postpunctuation) option of `refer`.

In some cases, it is necessary to override certain fields in a reference. For instance, each time a work is cited, you may want to specify different page numbers, and you may want to change certain fields. This citation will find the Lesk reference, but will add specific page numbers to the output, even though no page numbers appeared in the original reference.

```
.[
lesk inverted indexes
%P 7-13
%I Computing Services
%O UNX 12.2.2.
.]
```

The %I line will also override any previous publisher information, and the %O line will append some commentary. The `refer` program simply adds the new %P, %I, and %O strings to the output, and later strings definitions cancel earlier ones.

It is also possible to insert an entire citation that does not appear in the bibliographic database. This reference, for example, could be added as follows:

```
.[
%A Brian Kernighan
%T A troff Tutorial
%I Bell Laboratories
%D 1978
.]
```

This will cause `refer` to interpret the fields exactly as given, without searching the bibliographic database. This practice is not recommended, however, because it's better to add new references to the database, so they can be used again later.

If you want to change the way footnote numbers are printed, signals can be given on the `.[` and `.]` lines. For example, to say "See reference (2)," the citation should appear as:

```
See reference
.[(
partial citation
.]),
```

Note that blanks are significant on these signal lines. If a permanent change in the footnote format is desired, it is best to redefine the `[.` and `.]` strings.

5.10. Changing the `refer` Macros

This section is provided for those who wish to rewrite or modify the `refer` macros. This is necessary in order to make output correspond to specific journal requirements, or departmental standards. First there is an explanation of how new macros can be substituted for the old ones. Then several alterations are given as examples.

The `refer` macros for `nroff` and `troff` supplied by the **-ms** macro package reside in `/usr/lib/ms/ms.xref`; they are reference macros, for producing footnotes or endnotes. The `refer` macros used by `roffbib`, on the other hand, reside in `/usr/lib/tmac/tmac.bib`; they are for producing a stand-alone bibliography.

To change the macros used by `roffbib`, you will need to get your own version of this shell script into the directory where you are working. This command will get you a copy of `roffbib` and the macros it uses:

```
hostname% cp /usr/lib/tmac/tmac.bib bibmac
```

You can proceed to change bibmac as much as you like. Then when you use `roffbib`, you should specify your own version of the macros, which will be substituted for the normal ones

```
hostname% roffbib -m bibmac filename
```

where *filename* is the name of your bibliography file. Make sure there's a space between `-m` and `bibmac`.

If you want to modify the `refer` macros for use with `nroff` and the **-ms** macros, you will need to get a copy of "ms.ref":

```
hostname% cp /usr/lib/ms/ms.ref refmac
```

These macros are much like "bibmac", except they have `.FS` and `.FE` requests, to be used in conjunction with the **-ms** macros, rather than independently defined `.XP` and `.AP` requests. Now you can put this line at the top of the paper to be formatted:

```
.so refmac
```

Your new `refer` macros will override the definitions previously read in by the **-ms** package. This method works only if "refmac" is in the working directory.

Suppose you didn't like the way dates are printed, and wanted them to be parenthesized, with no comma before. There are five identical lines you will have to change. The first line below is the old way, while the second is the new way:

```
.if !"\\*([D"" , \\*([D\c
.if !"\\*([D"" \& (\\*([D)\c
```

In the first line, there is a comma and a space, but no parentheses. The "\c" at the end of each line indicates to `nroff` that it should continue, leaving no extra space in the output. The "\&" in the second line is the do-nothing character; when followed by a space, a space is sent to the output.

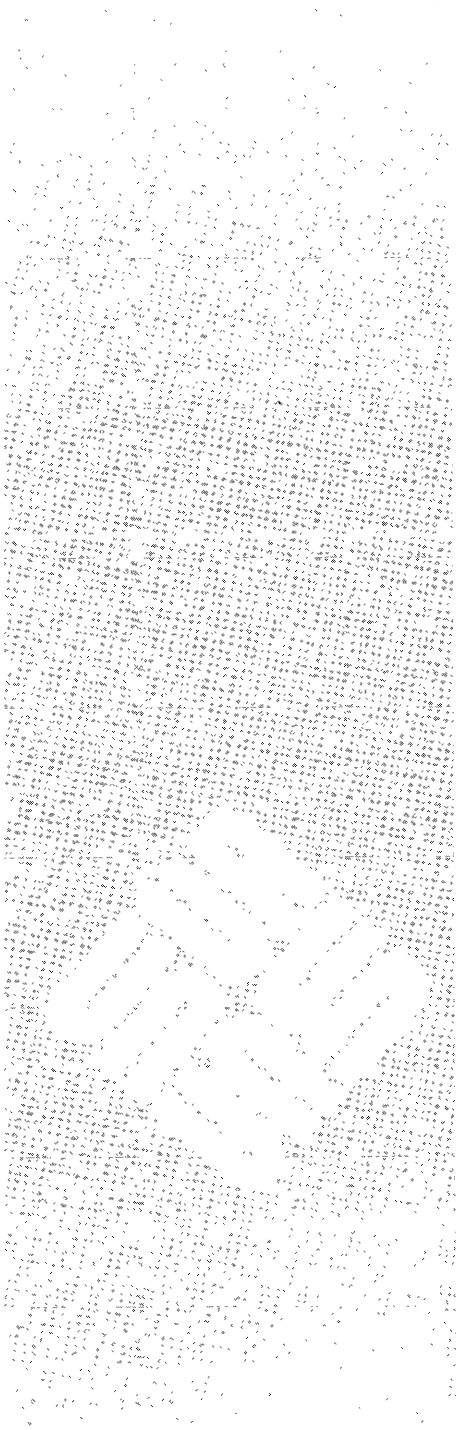
If you need to format a reference in the style favored by the Modern Language Association or Chicago University Press, in the form (city: publisher, date), then you will have to change the middle of the book macro [2 as follows:

```
\& (\c
.if !"\\*([C"" \\*([C:
\\*([I\c
.if !"\\*([D"" , \\*([D\c
)\c
```

This would print (Berkeley: Computing Services, 1982) if all three strings were present. The first line prints a space and a parenthesis; the second prints the city (and a colon) if present; the third always prints the publisher (books must have a publisher, or else they're classified as other); the fourth line prints a comma and the date if present; and the fifth line closes the parentheses. You would need to make similar changes to the other macros as well.

6

Formatting Tables with `tbl`



6

Formatting Tables with `tbl`

This chapter provides instructions for preparing `tbl` input to format tables and for running the `tbl` preprocessor on a file.[1] It also supplies numerous examples after which to pattern your own tables. The description of instructions is precise but technical, and the newcomer may prefer to glance over the examples first, as

`tbl` turns a simple description of a table into a `troff` or `nroff` program that prints the table. From now on, unless noted specifically, we'll refer to both `troff` and `nroff` as `troff` since `tbl` treats them the same. `tbl` makes phototypesetting tabular material relatively simple compared to normal typesetting methods. You may use `tbl` with the equation formatting program `eqn` or various layout macro packages, as `tbl` does not duplicate their functions.

Tables are made up of columns which may be independently centered, right-adjusted, left-adjusted, or aligned by decimal points. Headings may be placed over single columns or groups of columns. A table entry may contain equations, or may consist of several rows of text. Horizontal or vertical lines may be drawn as desired in the table, and any table or element may be enclosed in a box. For example:

1970 Federal Budget Transfers
(in billions of dollars)

State	Taxes collected	Money spent	Net
New York	22.91	21.35	–1.56
New Jersey	8.33	6.96	–1.37
Connecticut	4.12	3.10	–1.02
Maine	0.74	0.67	–0.07
California	22.29	22.42	+0.13
New Mexico	0.70	1.49	+0.79
Georgia	3.30	4.28	+0.98
Mississippi	1.15	2.32	+1.17
Texas	9.33	11.13	+1.80

[1] The material in this chapter is derived from *Tbl — A Program to Format Tables*, M.E. Lesk, Bell Laboratories, Murray Hill, New Jersey.

The input to `tbl` is text for a document, with the text preceded by a `.TS` (table start) command and followed by a `.TE` (table end) command. `tbl` processes the tables, generating `troff` formatting commands, and leaves the remainder of the text unchanged. The `.TS` and `.TE` lines are copied, too, so that `troff` page layout macros, such as the formatting macros, can use these lines to delimit and place tables as necessary. In particular, any arguments on the `.TS` or `.TE` lines are copied but otherwise ignored, and may be used by document layout macro commands.

The format of the input is as follows:

```
. . .
ordinary text of your document
. . .
.TS
first table
.TE
. . .
ordinary text of your document
. . .
.TS
second table
.TE
. . .
ordinary text of your document
. . .
```

where the format of each table is as follows:

```
.TS
options for the table ;
format describing the layout of the table .
data to be laid out in the table
                    .
                    .
                    .
data to be laid out in the table
.TE
```

Each table is independent, and must contain formatting information, indicated by *format describing the layout of the table*, followed by the *data to be laid out in the table*. You may precede the formatting information, which describes the individual columns and rows of the table, by *options for the table* that affect the entire table.

6.1. Running tbl

You can run tbl on a simple table by piping the tbl output to troff (or your installation's equivalent for the phototypesetter) with the command:

```
hostname% tbl file | troff -options
```

where *file* is the name of the file you want to format. For more complicated use, where there are several input files, and they contain equations and **-ms** macro package requests as well as tables, the normal command is:

```
hostname% tbl file1 file2... | eqn | troff -ms
```

You can, of course, use the usual options on the troff and eqn commands. The usage for nroff is similar to that for troff, but only printers such as the TELETYPE® Model 37 and Diablo-mechanism (DASI or GSI) or other printers that can handle reverse paper motions can print boxed tables directly. If you are running tbl on a line printer that does not filter reverse paper motions, use the col processor to filter the multicolumn output.

If you are using an IBM 1403 line printer without adequate driving tables or post-filters, there is a special –TX command line option to tbl which produces output that does not have fractional line motions in it. The only other command line options recognized by tbl are macro package specifications such as **-ms** and **-mm**. These options are turned into commands to fetch the corresponding macro files; usually it is more convenient to place these arguments on the troff part of the command line, tbl accepts them as well.

Caveats: Note that when you use eqn and tbl together on the same file, put tbl first. If there are no equations within tables, either order works, but it is usually faster to run tbl first, since eqn normally produces a larger expansion of the input than tbl. However, if there are equations within tables, using the delim mechanism in eqn, you must put tbl first or the output will be scrambled.

Also, beware of using equations in **n**-style columns; this is nearly always wrong, since tbl attempts to split numerical format items into two parts, and this is not possible with equations. To avoid this, use the delim(*xx*) table option to prevent splitting numerical columns within the delimiters.

For example, if the eqn delimiters are $$, giving delim($$) a numerical column such as 1245±16, means the column entry will not be divided after 1245, but after 16. This is the output: '1245±16' (all in one column within the table).

The only recommended in-line equation delimiters inside tables (tbl) are $$ or @@. Most of the other special characters have special meanings either inside eqn or tbl.

Some versions of tbl limit tables to twenty columns; however, use of more than 16 numerical columns may fail because of limits in troff, producing the 'too many number registers' message. Avoid using troff number registers used by tbl within tables; these include two-digit names from 31 to 99, and names of the forms #*x*, *x*+, *x* |, ^*x*, and *x*–, where *x* is any lower-case letter. The names ##,

#–, and #^ are also used in certain circumstances. To conserve number register names, the **n** and **a** formats share a register; hence the restriction that you may not use them in the same column.

For aid in writing layout macros, `tbl` defines a number register TW which is the table width; it is defined by the time that the `.TE` macro is invoked and may be used in the expansion of that macro. More importantly, 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 invoked at its end. Use of this macro in the page footer boxes a multi-page table. In particular, you can use the **-ms** macros to 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
```

a line of the form

```
.TH
```

must be given in the table after any table heading, or at the start if there aren't any. Material up to the `.TH` is placed at the top of each page of table; the remaining lines in the table are placed on several pages as required. For example:

```
.TS H
center box tab (/);
c s
l l .
Employees
_
Name/Phone
_
.TH
Jonathan Doe/123-4567
< etc. >
.TE
```

Note that this is *not* a feature of `tbl`, but of the **-ms** layout macros.

6.2. Input Commands

As indicated above, a table contains, first, global options, then a format section describing the layout of the table entries, and then the data to be printed. The format and data are always required, but not the options. The sections that follow explain how to enter the various parts of the table.

Options That Affect the Whole Table

There may be a single line of options affecting the whole table. If present, this line must follow the `.TS` line immediately, must contain a list of option names separated by spaces, tabs, or commas, and must be terminated by a semicolon. The allowable options are:

`center`	center the table (default is left-adjusted).
`expand`	make the table as wide as the current line length.
`box`	enclose the table in a box.
`allbox`	enclose each item in the table in a box.
`doublebox`	enclose the table in two boxes — a frame.
`tab(`*x*`)`	use *x* instead of tab to separate data items.
`linesize(`*n*`)`	set lines or rules (such as from `box`) in *n* point type.
`delim(`*xy*`)`	recognize *x* and *y* as the *eqn* delimiters.

A standard option line is:

```
center box tab (/) ;
```

which centers the table on the page, draws a box around it, and uses the slash '/' character as the column separator for data items.

The `tbl` program tries to keep boxed tables on one page by issuing appropriate `troff` 'need' (`.ne`) commands. These requests are calculated from the number of lines in the tables, so if there are spacing commands embedded in the input, these requests may be inaccurate. Use normal `troff` procedures, such as keep-release macros, in this case. If you must have a multi-page boxed table, use macros designed for the purpose, as explained above under *Running 'tbl'*.

Key Letters — Format Describing Data Items

The format section of the table specifies the layout of the columns. Each line in this section corresponds to one line of the table, except that the last line corresponds to all following lines up to the next `.T&`, if present as shown below. Each line contains a *key-letter* for each column of the table. It is good practice to separate the key letters for each column by spaces, tabs, or a visible character such as a slash '/'. Each key-letter is one of the following:

`L` or `l`	indicates a left-adjusted column entry.
`R` or `r`	indicates a right-adjusted column entry.
`C` or `c`	indicates a centered column entry.
`N` or `n`	indicates a numerical column entry, to line up the units digits of numerical entries.
`A` or `a`	indicates an alphabetic subcolumn; all corresponding entries are aligned on the left, and positioned so that the widest is centered within the column (see the "Some London Transport Statistics" example).

`S` or `s` indicates a spanned heading; that is, it indicates that the entry from the previous column continues across this column; not allowed for the first column.

indicates a vertically spanned heading; that is, it indicates that the entry from the previous row continues down through this row; not allowed for the first row of the table.

When you specify numerical alignment, `tbl` requires 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 no alignment is indicated, the item is centered in the column. However, you may use the special non-printing character string `\&` to override unconditionally dots and digits, or to align alphabetic data; this string lines up where a dot normally would, and then disappears from the final output. In the example below, the items shown at the left will be aligned in a numerical column as shown on the right:

```
13                  13
4.2                  4.2
26.4.12            26.4.12
abc                  abc
abc\&                abc
43\&3.22           433.22
749.12             749.12
```

Note: If numerical data are used in the same column with wider `L` or `r` type table entries, the widest *number* is centered relative to the wider `L` or `r` items (we use `L` here instead of `l` for readability; they have the same meaning as key-letters). Alignment within the numerical items is preserved. This is similar to the way `a` type data are formatted, as explained above. However, 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.

Note: Do not use the `n` and `a` items in the same column.

For readability, separate the key-letters describing each column with spaces. Indicate the end of the format section by a period. The layout of the key-letters in the format section resembles the layout of the actual data in the table. Thus a simple format is:

```
.TS
c  s  s
l  n  n .
text
.TE
```

which specifies a table of three columns. The first line of the table contains a centered heading that spans across all three columns; each remaining line contains a left-adjusted item in the first column followed by two columns of numerical data.

A sample table in this format is:

	Overall title	
Item-a	34.22	9.1
Item-b	12.65	.02
Items: c,d,e	23	5.8
Total	69.87	14.92

Optional Features of Key Letters

There may be extra information following a key-letter that modifies its basic behavior. Additional features of the key-letter system follow:

Horizontal lines
— A key-letter may be replaced by '_' (underscore) to indicate a horizontal line in place of the corresponding column entry, or by '=' to indicate a double horizontal line. You can also type this in the data portion. If an adjacent column contains a horizontal line, or if there are vertical lines adjoining this column, this horizontal line is extended to meet the nearby lines. If any data entry is provided for this column, it is ignored and a warning message is displayed.

Vertical lines
— A vertical bar may be placed between column key-letters. This draws a vertical line 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, a double vertical line is drawn.

Space between columns
— A number may follow the key-letter. This indicates the amount of separation between this column and the next column. The number normally specifies the separation in *ens* (one en is about the width of the letter 'n')[2]. If the `expand` option is used, these numbers are multiplied by a constant such that the table is as wide as the current line length. The default column separation number is 3. If the separation is changed, the largest space requested prevails.

Vertical spanning
— Normally, vertically-spanned items extending over several rows of the table are centered in their vertical range. If a key-letter is followed by `t` or `T`, any corresponding vertically-spanned item begins at the top line of its range.

Font changes
— A key-letter may be followed by a string containing a font name or number preceded by the letter `f` or `F`. This indicates that the corresponding column should be in a different font from the default font, which is usually Roman. All font names are one or two letters; a one-letter font name should be separated from whatever follows by a space or tab. The single letters `B`, `b`, `I`, and `i` are shorter synonyms for `fB` and `fI`. Font change commands

[2] More precisely, an *en* is a number of points (1 point = 1/72 inch) equal to half the current type size.

given with the table entries override these specifications.

Point size changes
— A key-letter may be followed by the letter `p` or `P` and a number to indicate the point size of the corresponding table entries. The number may be a signed digit, in which case it is taken as an increment or decrement from the current point size. If both a point size and a column separation value are given, one or more blanks must separate them.

Vertical spacing changes
— A key-letter may be followed by the letter `v` or `V` and a number to indicate the vertical line spacing to be used within a multi-line corresponding table entry. The number may be a signed digit, in which case it is taken as an increment or decrement from the current vertical spacing. A column separation value must be separated by blanks or some other specification from a vertical spacing request. This request has no effect unless the corresponding table entry is a text block (see *Text Blocks* below).

Column width indication
— A key-letter may be followed by the letter `w` or `W` and a width value in parentheses. This width is used as a minimum column width. If the largest element in the column is not as wide as the width value given after the `w`, the largest element is considered to be that wide. If the largest element in the column is wider than the specified value, its width is used. The width is also used as a default line length for included text blocks. Normal `troff` units can be used to scale the width value; if none is used, the default is ens. If the width specification is a unitless integer, you may omit the parentheses. If the width value is changed in a column, the *last* one given controls.

Equal width columns
— A key-letter may be followed by the letter `e` or `E` to indicate equal width columns. All columns whose key-letters are followed by `e` or `E` are made the same width. In this way, you can format a group of regularly spaced columns.

Note:
The order of the above features is immaterial; they need not be separated by spaces, except as indicated above 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)f I  6
```

Alternative notation
— Instead of listing the format of successive lines of a table on consecutive lines of the format section, separate successive line formats on the same line by commas. The format for the sample table above can be written:

```
c s s, l n n .
```

Default
— Column descriptors missing from the end of a format line are assumed to be L. The longest line in the format section, however, defines the number of columns in the table; extra columns in the data are ignored silently.

Data to be Formatted in the Table

Type the data for the table after the format line. Normally, each table line is typed as one line of data. Break very long input lines by typing a backslash '\' as a continuation marker at the end of the run-on line. That line is combined with the following line upon formatting and the '\' vanishes. The data for different columns, that is, the table entries, are separated by tabs, or by whatever character has been specified in the option *tabs* option. We recommend using a visible character such as the slash character '/'. There are a few special cases:

`troff` *commands within tables*
— An input line beginning with a '.' followed by anything except a digit is assumed to be a command to `troff` and is passed through unchanged, retaining its position in the table. So, for example, you can produce space within a table by `.sp` commands in the data.

Full width horizontal lines
— An input *line* containing only the character '_' (underscore) or '=' (equal sign) represents a single or double line, respectively, extending the full width of the *table.*

Single column horizontal lines
— An input table *entry* containing only the character '_' or '=' represents a single or double line extending the full width of the *column*. Such lines are extended to meet horizontal or vertical lines adjoining this column. To obtain these characters explicitly in a column, either precede them by '\&' or follow them by a space before the usual tab or newline.

Short horizontal lines
— An input table *entry* containing only the string '_' represents a single line as wide as the contents of the column. It is not extended to meet adjoining lines.

Vertically spanned items
— An input table entry containing only the character string `'\^'` indicates that the table entry immediately above spans downward over this row. It is equivalent to a table format key-letter of '^'.

Text blocks
— To include a block of text as a table entry, precede it by `T{` and follow it by `T}`. To enter, as a single entry in the table, something that cannot conveniently be typed as a simple string between tabs, use:

```
. . . T{
block of text
T} . . .
```

Note that the `T}` end delimiter must begin a line; additional columns of data may follow after a tab on the same line. See the 'New York Area Rocks' example for an illustration of included text blocks in a table. If you use more than twenty or thirty text blocks in a table, various limits in the `troff` program are likely to be exceeded, producing diagnostics such as `too many text block diversions`.

Text blocks are pulled out from the table, processed separately by `troff`, and replaced in the table as a solid block. If no line length is specified in the *block of text* itself, 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 number of columns in the table. The other parameters (point size, font, etc.) used in setting the *block of text* are those in effect at the beginning of the table (including the effect of the `.TS` macro) and any table format specifications of size, spacing and font, using the `p`, `v` and `f` modifiers to the column key-letters. Commands within the text block itself are also recognized, of course. However, `troff` commands within the table data but not within the text block do not affect that block.

Note:

Although you can put any number of lines in a table, only the first 200 lines are used in calculating the widths of the various columns. Arrange a multi-page table as several single-page tables if this proves to be a problem. Other difficulties with formatting may arise because, in the calculation of column widths all table entries are assumed to be in the font and size being used when the `.TS` command was encountered, except for font and size changes indicated (a) in the table format section and (b) within the table data (as in the entry `\s+3\fIdata\fP\s0`). Therefore, although arbitrary `troff` requests may be sprinkled in a table, use requests such as `.ps` (set the point size) with care to avoid confusing the width calculations.

Changing the Format of a Table

If you must change the format of a table after many similar lines, as with sub-headings or summarizations, use the `.T&` (table continue) command to change column parameters. The outline of such a table input is:

```
.TS                          start of the table
options afecting the whole table    ;
format of the columns    .
data to be formatted in the table
.
.
.
data to be formatted in the table
.T&                          indicates a new format for the table
format of the columns    .
data to be formatted in the table
.
.
.
data to be formatted in the table
.T&                          indicates a new format for the table
format of the columns    .
data to be formatted in the table
.
.
.
data to be formatted in the table
.TE                          end of the table
```

as in the 'Composition of Foods' and 'Some London Transport Statistics' examples. Using this procedure, each table line can be close to its corresponding format line.

Note: 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 width.

6.3. Examples

Here are some examples illustrating features of `tbl`. Glance through them to find one that you can adapt to your needs.

Although you can use a tab to separate columns of data, a visible character is easier to read. The standard column separator here is the slash (/). If a slash is part of the data, we indicate a different separator, as in the first example.

Input:

```
.TS
tab (%) box ;
c c c
l l l .
Language%Authors%Runs on

Fortran%Many%Almost anything
PL/1%IBM%360/370
C%BTL%11/45,H6000,370
BLISS%Carnegie-Mellon%PDP-10,11
IDS%Honeywell%H6000
Pascal%Stanford%370
.TE
```

Output:

Language	Authors	Runs on
Fortran	Many	Almost anything
PL/1	IBM	360/370
C	BTL	11/45,H6000,370
BLISS	Carnegie-Mellon	PDP-10,11
IDS	Honeywell	H6000
Pascal	Stanford	370

Input:

```
.TS
tab (/) allbox;
c s s
c c c
n n n .
AT&T Common Stock
Year/Price/Dividend
1971/41-54/$2.60
2/41-54/2.70
3/46-55/2.87
4/40-53/3.24
5/45-52/3.40
6/51-59/.95*
.TE
* (first quarter only)
```

Output:

AT&T Common Stock		
Year	Price	Dividend
1971	41-54	$2.60
2	41-54	2.70
3	46-55	2.87
4	40-53	3.24
5	45-52	3.40
6	51-59	.95*

* (first quarter only)

Input:

```
.TS
tab (/) box;
c s s
c | c | c
l | l | n .
Major New York Bridges
=
Bridge/Designer/Length
_
Brooklyn/J. A. Roebling/1595
Manhattan/G. Lindenthal/1470
Williamsburg/L. L. Buck/1600
_
Queensborough/Palmer &/1182
/  Hornbostel
_
//1380
Triborough/O. H. Ammann/_
//383
_
Bronx Whitestone/O. H. Ammann/2300
Throgs Neck/O. H. Ammann/1800
_
George Washington/O. H. Ammann/3500
.TE
```

Output:

Major New York Bridges		
Bridge	Designer	Length
Brooklyn	J. A. Roebling	1595
Manhattan	G. Lindenthal	1470
Williamsburg	L. L. Buck	1600
Queensborough	Palmer & Hornbostel	1182
Triborough	O. H. Ammann	1380
		383
Bronx Whitestone	O. H. Ammann	2300
Throgs Neck	O. H. Ammann	1800
George Washington	O. H. Ammann	3500

Input:

```
.TS
tab (/) ;
c c
np-2 | n | .
/Stack
/_
1/46
/_
2/23
/_
3/15
/_
4/6.5
/_
5/2.1
/_
.TE
```

Output:

	Stack
1	46
2	23
3	15
4	6.5
5	2.1

Input:

```
.TS
tab (/) box;
L L L
L L _
L L | LB
L L _
L L L .
january/february/march
april/may
june/july/Months
august/september
october/november/december
.TE
```

Output:

january	february	march
april	may	
june	july	Months
august	september	
october	november	december

Input:

```
.TS
 tab (/) box;
cfB s s s .
Composition of Foods
_
.T&
c  | c s s
c  | c s s
c   | c  | c  | c .
Food/Percent by Weight
\^/_
\^/Protein/Fat/Carbo-
\^/\^/\^/hydrate
_
.T&
l  | n  | n  | n .
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
.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

Input:

```
.TS
tab (/) allbox;
cfI s s
c cw(1i) cw(1i)
lp9 lp9 lp9 .
New York Area Rocks
Era/Formation/Age (years)
Precambrian/Reading Prong/>1 billion
Paleozoic/Manhattan Prong/400 million
Mesozoic/T{
.na
Newark Basin, incl.
Stockton, Lockatong, and Brunswick
formations; also Watchungs
and Palisades.
T}/200 million
Cenozoic/Coastal Plain/T{
On Long Island 30,000 years;
Cretaceous sediments redeposited
by recent glaciation.
.ad
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; also Watchungs and Palisades.	200 million
Cenozoic	Coastal Plain	On Long Island 30,000 years; Cretaceous sediments redeposited by recent glaciation.

Input:

```
.EQ
delim $$
.EN

. . .

.TS
tab (/) doublebox ;
c c
l l .
Name/Definition
.sp
.vs +2p
Gamma/$GAMMA (z) = int sub 0 sup inf  t sup {z-1} e sup -t dt$
Sine/$sin (x) = 1 over 2i ( e sup ix - e sup -ix )$
Error/$ roman erf (z) = 2 over sqrt pi int sub 0 sup z e sup {-t sup 2} dt$
Bessel/$ J sub 0 (z) = 1 over pi int sub 0 sup pi cos ( z sin theta ) d theta$
Zeta/$ zeta (s) = sum from k=1 to inf k sup -s ~~( Re~s > 1)$
.vs -2p
.TE
```

Output:

Name	Definition
Gamma	$\Gamma(z)=\int_0^\infty t^{z-1}e^{-t}\,dt$
Sine	$\sin(x)=\frac{1}{2i}(e^{ix}-e^{-ix})$
Error	$\mathrm{erf}(z)=\frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}dt$
Bessel	$J_0(z)=\frac{1}{\pi}\int_0^\pi \cos(z\sin\theta)d\theta$
Zeta	$\zeta(s)=\sum_{k=1}^{\infty}k^{-s}\quad(\mathrm{Re}\,s>1)$

Input:

```
.TS
box, tab (:) ;
cb s s s s
cp-2 s s s s
c | | c | c | c | c
c | | c | c | c | c
r2 | | n2 | n2 | n2 | n .
Readability of Text
Line Width & Leading for 10-Pt. Type
=
Line:Set:1-Point:2-Point:4-Point
Width:Solid:Leading:Leading:Leading
_
9 Pica:\-9.3:\-6.0:\-5.3:\-7.1
14 Pica:\-4.5:\-0.6:\-0.3:\-1.7
19 Pica:\-5.0:\-5.1: 0.0:\-2.0
31 Pica:\-3.7:\-3.8:\-2.4:\-3.6
43 Pica:\-9.1:\-9.0:\-5.9:\-8.8
.TE
```

Output:

Readability of Text
Line Width & Leading for 10-Pt. Type

Line Width	Set Solid	1-Point Leading	2-Point Leading	4-Point Leading
9 Pica	–9.3	–6.0	–5.3	–7.1
14 Pica	–4.5	–0.6	–0.3	–1.7
19 Pica	–5.0	–5.1	0.0	–2.0
31 Pica	–3.7	–3.8	–2.4	–3.6
43 Pica	–9.1	–9.0	–5.9	–8.8

Input:

```
.TS
tab (/) ;
c s
cip-2 s
l n
a n .
Some London Transport Statistics
(Year 1964)
Railway route miles/244
Tube/66
Sub-surface/22
Surface/156
.sp .5
.T&
l r
a r .
Passenger traffic \- railway
Journeys/674 million
Average length/4.55 miles
Passenger miles/3,066 million
.T&
l r
a r .
Passenger traffic \- road
Journeys/2,252 million
Average length/2.26 miles
Passenger miles/5,094 million
.T&
l n
a n .
.sp .5
Vehicles/12,521
Railway motor cars/2,905
Railway trailer cars/1,269
Total railway/4,174
Omnibuses/8,347
.T&
l n
a n .
.sp .5
Staff/73,739
Administrative, etc./8,553
Civil engineering/5,134
Electrical eng./1,714
Mech. eng. \- railway/4,310
Mech. eng. \- road/9,152
Railway operations/8,930
Road operations/35,946
.TE
```

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
Staff	73,739
Administrative, etc.	5,553
Civil engineering	5,134
Electrical eng.	1,714
Mech. eng. – railway	4,310
Mech. eng. – road	9,152
Railway operations	8,930
Road operations	35,946

Input:

```
.ps 8
.vs 10p
.TS
tab (/) center box;
c s s
ci s s
c c c
lB l n .
New Jersey Representatives
(Democrats)
.sp .5
Name/Office address/Phone
.sp .5
James J. Florio/23 S. White Horse Pike, Somerdale 08083/609-627-8222
William J. Hughes/2920 Atlantic Ave., Atlantic City 08401/609-345-4844
James J. Howard/801 Bangs Ave., Asbury Park 07712/201-774-1600
Frank Thompson, Jr./10 Rutgers Pl., Trenton 08618/609-599-1619
Andrew Maguire/115 W. Passaic St., Rochelle Park 07662/201-843-0240
Robert A. Roe/U.S.P.O., 194 Ward St., Paterson 07510/201-523-5152
Henry Helstoski/666 Paterson Ave., East Rutherford 07073/201-939-9090
Peter W. Rodino, Jr./Suite 1435A, 970 Broad St., Newark 07102/201-645-3213
Joseph G. Minish/308 Main St., Orange 07050/201-645-6363
Helen S. Meyner/32 Bridge St., Lambertville 08530/609-397-1830
Dominick V. Daniels/895 Bergen Ave., Jersey City 07306/201-659-7700
Edward J. Patten/Natl. Bank Bldg., Perth Amboy 08861/201-826-4610
.sp .5
.T&
ci s s
lB l n .
(Republicans)
.sp .5v
Millicent Fenwick/41 N. Bridge St., Somerville 08876/201-722-8200
Edwin B. Forsythe/301 Mill St., Moorestown 08057/609-235-6622
Matthew J. Rinaldo/1961 Morris Ave., Union 07083/201-687-4235
.TE
.ps 10
.vs 12p
```

Output:

New Jersey Representatives
(Democrats)

Name	Office address	Phone
James J. Florio	23 S. White Horse Pike, Somerdale 08083	609-627-8222
William J. Hughes	2920 Atlantic Ave., Atlantic City 08401	609-345-4844
James J. Howard	801 Bangs Ave., Asbury Park 07712	201-774-1600
Frank Thompson, Jr.	10 Rutgers Pl., Trenton 08618	609-599-1619
Andrew Maguire	115 W. Passaic St., Rochelle Park 07662	201-843-0240
Robert A. Roe	U.S.P.O., 194 Ward St., Paterson 07510	201-523-5152
Henry Helstoski	666 Paterson Ave., East Rutherford 07073	201-939-9090
Peter W. Rodino, Jr.	Suite 1435A, 970 Broad St., Newark 07102	201-645-3213
Joseph G. Minish	308 Main St., Orange 07050	201-645-6363
Helen S. Meyner	32 Bridge St., Lambertville 08530	609-397-1830
Dominick V. Daniels	895 Bergen Ave., Jersey City 07306	201-659-7700
Edward J. Patten	Natl. Bank Bldg., Perth Amboy 08861	201-826-4610
	(Republicans)	
Millicent Fenwick	41 N. Bridge St., Somerville 08876	201-722-8200
Edwin B. Forsythe	301 Mill St., Moorestown 08057	609-235-6622
Matthew J. Rinaldo	1961 Morris Ave., Union 07083	201-687-4235

This is a paragraph of normal text placed here only to indicate where the left and right margins are. Examine the appearance of centered tables or expanded tables, and observe how such tables are formatted.

Input:

```
.TS
center tab (/) ;
c s s s
c s s s
c c c c
n n n n .
LYKE WAKE WALK
Successful Crossings 1959–1966
Year/First Crossings/Repeats/Total
1959/89/23/112
1960/222/33/255
1961/650/150/800
1962/1100/267/1367
1963/1054/409/1463
1964/1413/592/2005
1965/2042/771/2813
1966/2537/723/3260
.TE
```

Output:

LYKE WAKE WALK
Successful Crossings 1959–1966

Year	First Crossings	Repeats	Total
1959	89	23	112
1960	222	33	255
1961	650	150	800
1962	1100	267	1367
1963	1054	409	1463
1964	1413	592	2005
1965	2042	771	2813
1966	2537	723	3260

Input:

```
.TS
tab (/) box;
cb   s   s   s
c | c | c   s
ltiw(1i) | ltw(2i) | lp8 | lw(1.6i)p8 .
Some Interesting Places
_
Name/Description/Practical Information
_
T{
American Museum of Natural History
T}/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}/Hours/10-5, ex. Sun 11-5, Wed. to 9
\^/\^/Location/T{
Central Park West & 79th St.
T}
\^/\^/Admission/Donation: $1.00 asked
\^/\^/Subway/AA to 81st St.
\^/\^/Telephone/212-873-4225
_
Bronx Zoo/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}/Hours/T{
10-4:30 winter, to 5:00 summer
T}
\^/\^/Location/T{
185th St. & Southern Blvd, the Bronx.
T}
\^/\^/Admission/$1.00, but Tu,We,Th free
\^/\^/Subway/2, 5 to East Tremont Ave.
\^/\^/Telephone/212-933-1759
_
Brooklyn Museum/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}/Hours/Wed-Sat, 10-5, Sun 12-5
\^/\^/Location/T{
Eastern Parkway & Washington Ave., Brooklyn.
T}
\^/\^/Admission/Free
\^/\^/Subway/2,3 to Eastern Parkway.
\^/\^/Telephone/212-638-5000
_
T{
New-York Historical Society
T}/T{
All the original paintings for Audubon's
.I
Birds of America
.R
are here, as are exhibits of American decorative arts, New York history,
Hudson River school paintings, carriages, and glass paperweights.
T}/Hours/T{
Tues-Fri & Sun, 1-5; Sat 10-5
T}
\^/\^/Location/T{
Central Park West & 77th St.
T}
\^/\^/Admission/Free
\^/\^/Subway/AA to 81st St.
\^/\^/Telephone/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 Parkway & 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

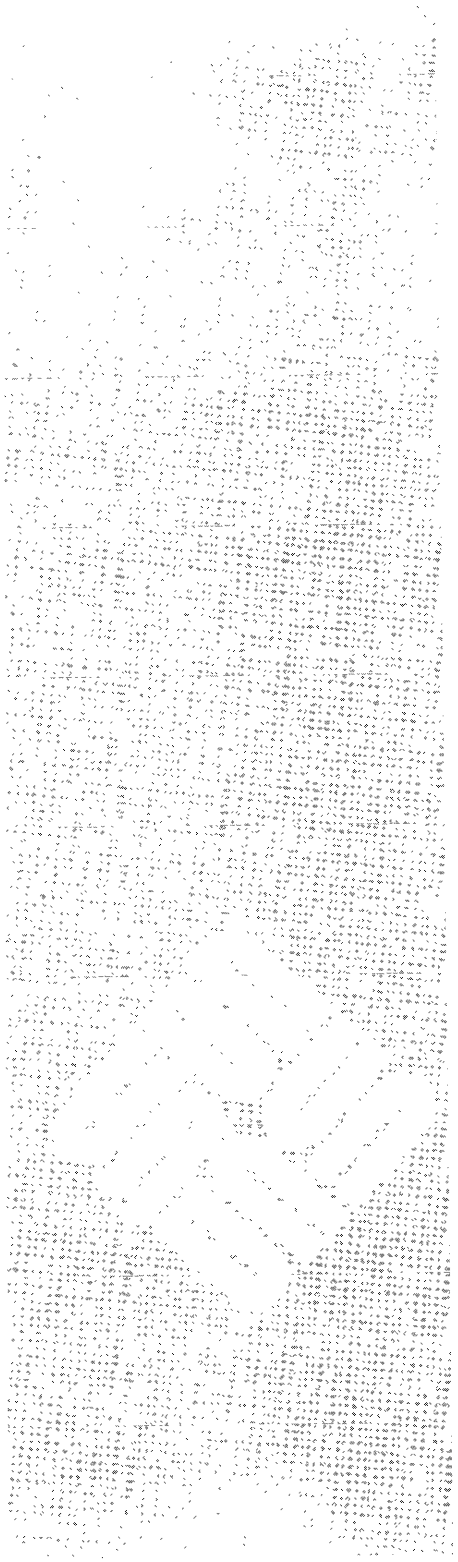
6.4. `tbl` Commands

Table 6-1 `tbl` *Command Characters and Words*

Command	*Meaning*
`a A`	Alphabetic subcolumn
`allbox`	Draw box around all items
`b B`	Boldface item
`box`	Draw box around table
`c C`	Centered column
`center`	Center table in page
`doublebox`	Doubled box around table
`e E`	Equal width columns
`expand`	Make table full line width
`f F`	Font change
`i I`	Italic item
`l L`	Left adjusted column
`n N`	Numerical column
nnn	Column separation
`p P`	Point size change
`r R`	Right adjusted column
`s S`	Spanned item
`t T`	Vertical spanning at top
`tab (`*x*`)`	Change data separator character
`T{  T}`	Text block
`v V`	Vertical spacing change
`w W`	Minimum width value
`.`*xx*	Included `troff` command
|	Vertical line
| |	Double vertical line
`^`	Vertical span
`\^`	Vertical span
`=`	Double horizontal line
`_`	Horizontal line
`\_`	Short horizontal line

7

Typesetting Mathematics with eqn



7

Typesetting Mathematics with eqn

This chapter explains how to use the eqn preprocessor for printing mathematics on a phototypesetter, and provides numerous examples after which to model equations in your documents.[1]

You describe mathematical expressions in an English-like language that the eqn program translates into troff commands for final troff formatting. In other words, eqn sets the mathematics while troff does the body of the text. eqn provides accurate and relatively easy mathematical phototypesetting, which is not easy to accomplish with normal typesetting machines. Because the mathematical expressions are embedded in the running text of a manuscript, the entire document is produced in one process. For example, you can set in-line expressions like $\lim_{x \to \pi/2} (\tan x)^{\sin 2x} = 1$ or display equations like

$$G(z) = e^{\ln G(z)} = \exp\left[\sum_{k \geq 1} \frac{S_k z^k}{k}\right] = \prod_{k \geq 1} e^{S_k z^k / k}$$

$$= \left[1 + S_1 z + \frac{S_1^2 z^2}{2!} + \cdots\right]\left[1 + \frac{S_2 z^2}{2} + \frac{S_2^2 z^4}{2^2 \cdot 2!} + \cdots\right] \cdots$$

$$= \sum_{m \geq 0} \left[\sum_{\substack{k_1, k_2, \ldots, k_m \geq 0 \\ k_1 + 2k_2 + \cdots + mk_m = m}} \frac{S_1^{k_1}}{1^{k_1} k_1!} \frac{S_2^{k_2}}{2^{k_2} k_2!} \cdots \frac{S_m^{k_m}}{m^{k_m} k_m!} \right] z^m$$

eqn knows relatively little about mathematics. In particular, mathematical symbols like +, −, ×, parentheses, and so on have no special meanings. eqn is quite happy to set these symbols, and they will look good.

eqn also produces mathematics with nroff. The input is identical, but you have to use the programs neqn and nroff instead of eqn and troff. Of course, some things won't look as good because your workstation or terminal does not provide the variety of characters, sizes and fonts that a phototypesetter does, but the output is usually adequate for proofreading.

---

[1] The material in this chapter is derived from *A System for Typesetting Mathematics*, B.W. Kernighan, L. L. Cherry and *Typesetting Mathematics — User's Guide*, B.W. Kernighan, L.L. Cherry, Bell Laboratories, Murray Hill, New Jersey.

7.1. Displaying Equations — .EQ and .EN

To tell eqn where a mathematical expression begins and ends, mark it with lines beginning .EQ and .EN. Thus if you type the lines:

```
.EQ
x=y+z
.EN
```

your output will look like:

$$x=y+z$$

eqn copies '.EQ' and '.EN' through untouched. This means that you have to take care of things like centering, numbering, and so on yourself. The common way is to use the troff and nroff macro package package '**-ms**', which provides macros for centering, indenting, left-justifying and making numbered equations.

With the **-ms** package, equations are centered by default. To left-justify an equation, use .EQ L instead of .EQ. To indent it, use .EQ I.

You can also supplement eqn with troff commands as desired; for example, you can produce a centered display with the input:

```
.ce
.EQ
x sub i = y sub i ...
.EN
```

which produces

$$x_i=y_i \cdots$$

You can call out any of these by an arbitrary 'equation number,' which will be placed at the right margin. For example, the input

```
.EQ I (3.1a)
x = f(y/2) + y/2
.EN
```

produces the output

$$x=f(y/2)+y/2 \qquad (3.1a)$$

There is also a shorthand notation so you can enter in-line expressions like π_i^2 without .EQ and .EN. This is described in the section "Shorthand for In-line Equations."

7.2. Running eqn and neqn

To print a document that contains mathematics on the phototypesetter, use:

```
hostname% eqn files | troff -options | lpr -t -Pprinter
```

troff or your installation's equivalent does the formatting, which is sent to your phototypesetter as indicated by -P*printer*. If you use the **-ms** macro package for example, type:

```
hostname% eqn files | troff -ms -t | lpr -t -Pprinter
```

To display equations on the standard output, your workstation screen, use nroff as follows:

```
hostname% neqn files | nroff -options
```

The language for equations recognized by neqn is identical to that of eqn, although of course the output is more restricted. You can use the online rendition of the mathematical formulae for proofing, but the output does not accurately represent the symbols and fonts. You can of course pipe the output through more for easier viewing:

```
hostname% neqn files | nroff -options | more
```

or redirect it to a file:

```
hostname% neqn files | nroff -options > newfile
```

To use a GSI or DASI terminal as the output device, type:

```
hostname% neqn files | nroff -Tx
```

where *x* is the terminal type you are using, such as *300* or *300S*. To send neqn output to the printer, type:

```
hostname% neqn file | nroff -options | lpr -Pprinter
```

You can use eqn and neqn with the tbl program for setting tables that contain mathematics. Use tbl before eqn or neqn, like this:

```
hostname% tbl files | eqn | troff -options
```

or

```
hostname% tbl files | neqn | nroff -options
```

7.3. Putting Spaces in the Input Text

eqn removes spaces and newlines within an expression and leaves normal text alone. Thus, between .EQ and .EN,

```
.EQ
x=y+z
.EN
```

and

```
.EQ
x = y + z
.EN
```

and

```
.EQ
x    =    y
     + z
.EN
```

all produce the same output, namely:

$$x=y+z$$

You should use spaces and newlines freely to make your input equations readable and easy to edit. In particular, very long lines are a bad idea, since they are often hard to fix if you make a mistake.

The only way eqn can deduce that some sequence of letters might be special is if that sequence is separated from the letters on either side of it. To do this, surround a special word by ordinary spaces (or tabs or newlines), as shown in the previous section.

You can also make special words stand out by surrounding them with tildes or circumflexes:

```
.EQ
x~=~2~pi~int~sin~(~omega~t~)~dt
.EN
```

is much the same as the last example, except that the tildes not only separate the magic words like sin, omega, and so on, but also add extra spaces, one space per tilde:

$$x = 2 \pi \int \sin (\omega t) \, dt$$

You can also use braces { } and double quotes " . . . " to separate special words; these characters that have special meanings are described later.

Remembering that a blank is a delimiter can be a problem. For instance, a common mistake is typing:

```
.EQ
f(x sub i)
.EN
```

which produces

$$f(x_{i)}$$

instead of

$$f(x_i)$$

eqn cannot tell that the right parenthesis is not part of the subscript. Type instead:

```
.EQ
f(x sub i )
.EN
```

7.4. Producing Spaces in the Output Text

To force extra spaces into the output, use a tilde ~ for each space you want:

```
.EQ
x~=~y~+~z
.EN
```

gives

$$x = y + z$$

You can also use a circumflex ^, which gives a space half the width of a tilde. It is mainly useful for fine-tuning. Use tabs to position pieces of an expression, but you must use troff commands to set the tab stops.

7.5. Symbols, Special Names, and Greek Letters

eqn knows some mathematical symbols, some mathematical names, and the Greek alphabet. For example,

```
.EQ
x=2 pi int sin ( omega t)dt
.EN
```

produces

$$x=2\pi\int\sin(\omega t)dt$$

Here the spaces in the input are **necessary** to tell eqn that int, pi, sin, and omega are separate entities that should get special treatment. The sin, digit 2, and parentheses are set in roman type instead of italic; pi and omega are made Greek; and int becomes the integral sign.

When in doubt, leave spaces around separate parts of the input. A *very* common error is to type

```
f(pi)
```

without leaving spaces on both sides of the `pi`. As a result, `eqn` does not recognize `pi` as a special word, and it appears as $f(pi)$ instead of $f(\pi)$.

A complete list of `eqn` names appears in the section "Precedences and Keywords." You can also use special characters available in `troff` for anything `eqn` doesn't know about.

7.6. Subscripts and Superscripts — `sub` and `sup`

To obtain subscripts and superscripts, use the words `sub` and `sup`.

```
.EQ
x sup 2 + y sub k
.EN
```

gives

$$x^2+y_k$$

`eqn` takes care of all the size changes and vertical motions needed to make the output look right. You must surround the words `sub` and `sup` by spaces; *x sub2* gives you *xsub*2 instead of x_2. As another example, consider:

```
.EQ
x sup 2 + y sup 2 = z sup 2
.EN
```

which produces:

$$x^2+y^2=z^2$$

Furthermore, don't forget to leave a space (or a tilde, etc.) to mark the end of a subscript or superscript. A common error is to say something like

```
.EQ
y = (x sup 2)+1
.EN
```

which causes

$$y=(x^{2)+1}$$

instead of the intended

$$y=(x^2)+1$$

which is produced by:

```
.EQ
y = (x sup 2 )+1
.EN
```

Subscripted subscripts and superscripted superscripts also work:

```
.EQ
x sub i sub 1
.EN
```

is

$$x_{i_1}$$

A subscript and superscript on the same thing are printed one above the other if the subscript comes first:

```
.EQ
x sub i sup 2
.EN
```

is

$$x_i^2$$

Other than this special case, `sub` and `sup` group to the right, so

```
x sup y sub z
```

means x^{y_z}, not x^{y}_{z}.

7.7. Grouping Equation Parts — { and }

Normally, the end of a subscript or superscript is marked simply by a blank, tab, tilde, and so on. If the subscript or superscript is something that has to be typed with blanks in it, use the braces { and } to mark the beginning and end of the subscript or superscript:

```
.EQ
e sup {i omega t}
.EN
```

is

$$e^{i\omega t}$$

You can *always* use braces to force `eqn` to treat something as a unit, or just to make your intent perfectly clear. Thus:

```
.EQ
x sub {i sub 1} sup 2
.EN
```

is

$$x_{i_1}^2$$

with braces, but

```
.EQ
x sub i sub 1 sup 2
.EN
```

is

$$x_{i_1^2}$$

which is rather different.

Braces can occur within braces if necessary:

```
.EQ
e sup {i pi sup {rho +1}}
.EN
```

is

$$e^{i\pi^{\rho+1}}$$

The general rule is that anywhere you could use some single entry like x, you can use an arbitrarily complicated entry if you enclose it in braces. `eqn` looks after all the details of positioning it and making it the right size.

In all cases, make sure you have the right number of braces. Leaving one out or adding an extra causes `eqn` to complain bitterly.

Occasionally you have to print braces. To do this, enclose them in double quotes, like `"{"`. Quoting is discussed in more detail in *Quoted Text.*

7.8. Fractions — `over`

To make a fraction, use the word `over`:

```
.EQ
a+b over 2c =1
.EN
```

gives

$$\frac{a+b}{2c}=1$$

The line is made the right length and positioned automatically.

```
.EQ
a+b over c+d+e = 1
.EN
```

produces

$$\frac{a+b}{c+d+e}=1$$

Use braces to clarify what goes over what:

```
.EQ
{alpha + beta} over {sin (x)}
.EN
```

is

$$\frac{\alpha+\beta}{\sin(x)}$$

When there is both an `over` and a `sup` in the same expression, `eqn` does the `sup` before the `over`, so

```
.EQ
-b sup 2 over pi
.EN
```

is $\frac{-b^2}{\pi}$ instead of $-b^{\frac{2}{\pi}}$ The rules that determine which operation is done first in cases like this are summarized in the section "Precedences and Keywords." When in doubt, however, use braces to make clear what goes with what.

7.9. Square Roots — `sqrt`

To draw a square root, use `sqrt`:

```
.EQ
sqrt a+b
.EN
```

produces

$$\sqrt{a+b}$$

and

```
.EQ
sqrt a+b + 1 over sqrt {ax sup 2 +bx+c}
.EN
```

is

$$\sqrt{a+b}+\frac{1}{\sqrt{ax^2+bx+c}}$$

Note: Square roots of tall quantities look sloppy because a root-sign big enough to cover the quantity is too dark and heavy:

```
.EQ
sqrt {a sup 2 over b sub 2}
.EN
```

is

$$\sqrt{\frac{a^2}{b_2}}$$

Big square roots are generally better written as something to a power:

$$(a^2/b_2)^{\frac{1}{2}}$$

which is

```
.EQ
(a sup 2 /b sub 2 ) sup {1 over 2}
.EN
```

7.10. Summation, Integral, and Other Large Operators

To produce summations, integrals, and similar constructions, use:

```
.EQ
sum from i=0 to {i= inf} x sub i
.EN
```

which produces

$$\sum_{i=0}^{i=\infty} x_i$$

Notice that you use braces to indicate where the upper part $i=\infty$ begins and ends. No braces are necessary for the lower part $i=0$, because it does not contain any blanks. The braces will never hurt, and if the `from` and `to` parts contain any blanks, you must use braces around them.

The `from` and `to` parts are both optional, but if both are used, they have to occur in that order.

Other useful characters can replace the `sum` in our example:

```
.EQ
int   prod   union   inter
.EN
```

become, respectively,

$$\int \quad \prod \quad \cup \quad \cap$$

Since the thing before the `from` can be anything, even something in braces, `from-to` can often be used in unexpected ways:

```
.EQ
lim from {n -> inf} x sub n =0
.EN
```

is

$$\lim_{n \to \infty} x_n = 0$$

7.11. Size and Font Changes

By default, equations are set in 10-point type with standard mathematical conventions to determine what characters are in roman and what in italic. Although eqn makes a valiant attempt to use aesthetically pleasing sizes and fonts, it is not perfect. To change sizes and fonts, use `size` *n* and `roman`, `italic`, `bold` and `fat`. Like `sub` and `sup`, size and font changes affect only the thing that follows them; they revert to the normal situation at the end of it. Thus

```
.EQ
bold x y
.EN
```

is

$$\mathbf{x}\,y$$

and

```
.EQ
size 14 bold x = y +
   size 14 {alpha + beta}
.EN
```

gives

$$\mathbf{x}=y+\alpha+\beta$$

As always, you can use braces if you want to affect something more complicated than a single letter. For example, you can change the size of an entire equation by

```
.EQ
size 12 { ... }
.EN
```

Legal sizes that may follow `size` are the same as those allowed in `troff`: 6, 7, 8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24, 28, 36. You can also change the size by a given increment or decrement. For example, you can say `size +2` to make the size two points bigger, or `size -3` to make it three points smaller. This is easier because you don't have to know what the current size is.

The `size` variable in eqn translates into a `troff` `\s` construct. However, `troff` only recognizes one digit after the + or – sign. Therefore, `\s+9` or `\s-9` are respectively the largest incremental and decremental point size changes.

If you are using fonts other than roman, italic and bold, you can say `font` *X* where *X* is a one character `troff` name or number for the font. Since eqn is

tuned for roman, italic and bold, other fonts may not appear quite as good.

The `fat` operation takes the current font and widens it by overstriking: `fat grad` is ∇ and `fat {x sub i}` is x_i.

If an entire document is to be in a non-standard size or font, it is a severe nuisance to have to write out a size and font change for each equation. Accordingly, you can set a global size or font which thereafter affects all equations. At the beginning of any equation, you might say, for instance,

```
.EQ
gsize 16
gfont R
 ...
.EN
```

to set the size to 16 and the font to roman thereafter. In place of `R`, you can use any of the `troff` font names. The size after `gsize` can be a relative change with + or –.

Generally, `gsize` and `gfont` will appear at the beginning of a document but they can also appear throughout a document: you can change the global font and size as often as needed. For example, in a footnote[2] you will typically want the size of equations to match the size of the footnote text, which is two points smaller than the main text. Don't forget to reset the global size at the end of the footnote.

7.12. Diacritical Marks

To get accent marks on top of letters, there are several words:

```
x dot       ẋ
x dotdot    ẍ
x hat       x̂
x tilde     x̃
x vec       x⃗
x dyad      x⃡
x bar       x̄
x under     x̲
```

The diacritical mark is placed at the right height. The `bar` and `under` are made the right length for the entire construct, as in $\overline{x+y+z}$; other marks are centered. For example

[2] Like this one, in which we have a few random expressions like x_i and π^2. The sizes for these were set by the command `gsize -2`.

```
.EQ
x dot under + x hat + y tilde
+ X hat + Y dotdot = z+Z bar
.EN
```

produces

$$\underline{\dot{x}}+\hat{x}+\tilde{y}+\hat{X}+\ddot{Y}=\overline{z+Z}$$

7.13. Quoted Text

Any input entirely within quotes ("...") is not subject to any of the font changes and spacing adjustments that you normally set. This provides a way to do your own spacing and adjusting if needed:

```
.EQ
italic "sin(x)" + sin (x)
.EN
```

is

$$sin(x)+\sin(x)$$

You also use quotes to get braces and other eqn keywords printed:

```
.EQ
"{ size alpha }"
.EN
```

is

$$\{ size\ alpha \}$$

and

```
.EQ
roman "{ size alpha }"
.EN
```

is

$$\{ \text{size alpha} \}$$

The construction " " is often used as a place-holder when grammatically eqn needs something, but you don't actually want anything in your output. For example, to make ^{2}He, you can't just type `sup 2 roman He` because a `sup` has to be a superscript *on* something. Thus you must say

```
.EQ
"" sup 2 roman He
.EN
```

To get a literal quote use \". troff characters like \(bs can appear unquoted, but more complicated things like horizontal and vertical motions with \h and \v should always be quoted.

7.14. Lining Up Equations — mark and lineup

Sometimes it's necessary to line up a series of equations at some horizontal position, often at an equals sign. To do this, use the two operations called mark and lineup.

The word mark may appear once at any place in an equation. It remembers the horizontal position where it appeared. Successive equations can contain one occurrence of the word lineup. The place where lineup appears is made to line up with the place marked by the previous mark if at all possible. Thus, for example, you can say

```
.EQ I
x+y mark = z
.EN
.EQ I
x lineup = 1
.EN
```

to produce

$$x+y=z$$

$$x=1$$

For reasons beyond the scope of this chapter, when you use eqn and **-ms**, use either .EQ I or .EQ L, as mark and lineup don't work with centered equations. Also bear in mind that mark doesn't look ahead;

```
.EQ
x mark =1
 ...
x+y lineup =z
.EN
```

isn't going to work, because there isn't room for the x+y part after the mark has processed the x.

7.15. Big Brackets

To get big brackets [ ], braces { }, parentheses ( ), and bars | | around things, use the left and right commands:

```
.EQ
left { a over b + 1 right }
 ~=~ left ( c over d right )
 + left [ e right ]
.EN
```

is

$$\left\{\frac{a}{b}+1\right\} = \left[\frac{c}{d}\right] + \left[e\right]$$

The resulting brackets are made big enough to cover whatever they enclose. Other characters can be used besides these, but they are not likely to look very good. Two exceptions are the `floor` and `ceiling` characters:

```
.EQ
left floor x over y right floor
<= left ceiling a over b right ceiling
.EN
```

produces

$$\left\lfloor\frac{x}{y}\right\rfloor \leq \left\lceil\frac{a}{b}\right\rceil$$

Several warnings about brackets are in order. First, braces are typically bigger than brackets and parentheses, because they are made up of three, five, seven, etc., pieces, while brackets can be made up of two, three, etc. Second, big left and right parentheses often look poor, because the character set is poorly designed.

The `right` part may be omitted: a 'left something' need not have a corresponding 'right something'. If the `right` part is omitted, put braces around the thing you want the left bracket to encompass. Otherwise, the resulting brackets may be too large.

If you want to omit the `left` part, things are more complicated, because technically you can't have a `right` without a corresponding `left`. Instead you have to say

```
left "" ..... right )
```

for example. The `left ""` means a 'left nothing'. This satisfies the rules without hurting your output.

7.16. Piles — `pile`

There is a general facility for making vertical piles of things; it comes in several flavors. For example:

```
.EQ
A ~=~ left [
  pile { a above b above c }
  ~~ pile { x above y above z }
right ]
.EN
```

will make

$$A = \left[\begin{matrix} a & x \\ b & y \\ c & z \end{matrix}\right]$$

The elements of the pile are centered one above another at the right height for most purposes. There can be as many elements as you want. The keyword `above` is used to separate the pieces; put braces around the entire list. The elements of a pile can be as complicated as needed, even containing more piles.

Three other forms of pile exist: `lpile` makes a pile with the elements left-justified; `rpile` makes a right-justified pile; and `cpile` makes a centered pile, just like `pile`. The vertical spacing between the pieces is somewhat larger for `lpiles`, `rpiles`, and `cpiles` than it is for ordinary piles. For example:

```
.EQ
roman sign (x)~=~
left {
   lpile {1 above 0 above -1}
   ~~ lpile
    {if~x>0 above if~x=0 above if~x<0}
.EN
```

makes

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x>0 \\ 0 & \text{if } x=0 \\ -1 & \text{if } x<0 \end{cases}$$

Notice the left brace without a matching right one.

7.17. Matrices — `matrix`

It is also possible to make matrices. For example, to make a neat array like

$$\begin{matrix} x_i & x^2 \\ y_i & y^2 \end{matrix}$$

you have to type

```
.EQ
matrix {
  ccol { x sub i above y sub i }
  ccol { x sup 2 above y sup 2 }
}
.EN
```

This produces a matrix with two centered columns. The elements of the columns are then listed just as for a pile, each element separated by the word `above`. You can also use `lcol` or `rcol` to left or right adjust columns. Each column can be separately adjusted, and there can be as many columns as you like.

The reason for using a matrix instead of two adjacent piles, by the way, is that if the elements of the piles don't all have the same height, they won't line up properly. A matrix forces them to line up, because it looks at the entire structure before deciding what spacing to use.

A word of warning about matrices: **each column must have the same number of elements in it.** Otherwise, results are unpredictable.

7.18. Shorthand for In-line Equations — `delim`

In a mathematical document, it is necessary to follow mathematical conventions not just in display equations, but also in the body of the text. For example you need variable names like x to be in italics. Although you can do this by surrounding the appropriate parts with the macro requests `.EQ` and `.EN`, the continual repetition of `.EQ` and `.EN` is a nuisance. Furthermore, with **-ms**, `.EQ` and `.EN` imply a displayed equation.

`eqn` provides a shorthand for short in-line expressions. You can define two characters to mark the left and right ends of an in-line equation, and then type expressions in the middle of text lines. To set both the left and right characters to dollar signs, for example, add to the beginning of your document the three lines

```
.EQ
delim $$
.EN
```

Having done this, you can then say things like

```
Let $alpha sub i$ be the primary variable, and let $beta$
be zero.  Then we can show that $x sub 1$ is $>=0$.
```

This works as you might expect; spaces, newlines, and so on are significant in the text, but not in the equation part itself. Multiple equations can occur in a single input line.

Enough room is left before and after a line that contains in-line expressions that something like $sum from i=1 to n x sub i$ does not interfere with the lines surrounding it.

The printed result looks like: Let α_i be the primary variable, and let β be zero. Then we can show that x_1 is ≥ 0.

To turn off the delimiters, use:

```
.EQ
delim off
.EN
```

Notes: Don't use braces, tildes, circumflexes, or double quotes as delimiters; chaos will result. Also, if you're using `tbl`, don't use sharps (pound signs) either.

7.19. Definitions — `define`

`eqn` provides a string-naming facility so you can give a frequently-used string of characters a name, and thereafter just type the name instead of the whole string. For example, if the sequence

```
.EQ
x sub i sub 1 + y sub i sub 1
.EN
```

appears repeatedly throughout a paper, you can save re-typing it each time by defining it like this:

```
.EQ
define  xy  'x sub i sub 1 + y sub i sub 1'
.EN
```

This makes `xy` a shorthand for whatever characters occur between the single quotes in the definition. You can use any character instead of quote to mark the ends of the definition, as long as it doesn't appear inside the definition.

Now you can use `xy` like this:

```
.EQ
f(x) = xy ...
.EN
```

and so on. Each occurrence of `xy` will expand into what it was defined as. Be sure to leave spaces or their equivalent around the name when you actually use it, so `eqn` will be able to identify it as special.

There are several things to watch out for. First, although definitions can use previous definitions, as in

```
.EQ
define  xi  ' x sub i '
define  xi1  ' xi sub 1 '
.EN
```

Don't define something in terms of itself. A common error is to say

```
.EQ
define  X  ' roman X '
.EN
```

This is a guaranteed disaster, since X is now defined in terms of itself. If you say

```
.EQ
define  X  ' roman "X" '
.EN
```

however, the quotes protect the second X, and everything works fine.

You can redefine `eqn` keywords. You can make slash (/) mean `over` by saying

```
.EQ
define  /  ' over '
.EN
```

or redefine `over` as / with

```
.EQ
define  over  ' / '
.EN
```

If you need things to print on a workstation or terminal as well as on the phototypesetter, it is sometimes worth defining a symbol differently in `neqn` and `eqn`. To do this, use `ndefine` and `tdefine`. A definition made with `ndefine` only takes effect if you are running `neqn`; if you use `tdefine`, the definition only applies for `eqn`. Names defined with plain `define` apply to both `eqn` and `neqn`.

7.20. Tuning the Spacing

Although `eqn` tries to get most things at the right place on the paper, it isn't perfect, and occasionally you will need to tune the output to make it just right. You can get small extra horizontal spaces with tilde and circumflex. You can also say `back` *n* and `fwd` *n* to move small amounts horizontally. The *n* is how far to move in 1/100s of an em (an em is about the width of the letter 'm'.) Thus `back 50` moves back about half the width of an m. Similarly you can move things up or down with `up` *n* and `down` *n*. As with `sub` or `sup`, the local motions affect the next thing in the input, and this can be anything if it is enclosed in braces.

7.21. Troubleshooting

If you make a mistake in an equation, like leaving out a brace, having one too many, or having a `sup` with nothing before it, `eqn` tells you with the message:

```
syntax error between lines x and y, file z
```

where *x* and *y* are approximately the lines between which the trouble occurred, and *z* is the name of the file in question. The line numbers are approximate, so look nearby as well. There are also self-explanatory messages that arise if you leave out a quote or try to run `eqn` on a non-existent file.

If you want to check a document before actually printing it, run:

```
hostname% eqn files >/dev/null
```

to throw away the output but display the messages.

If you use something like dollar signs as delimiters, it is easy to leave one out. You may also occasionally forget one half of a pair of macros or have an unbalanced font change. These can cause problems, but you can check for balanced pairs of delimiters and macros with `checkeq` and `checknr`. For instance, to run `checkeq` on this chapter called `eqn.ug` to check for unbalanced pairs of `.EQ`s and `.EN`s, type:

```
hostname% checkeq eqn.ug
eqn.ug:
   New delims , line 2
    in EQ, line 2
   Spurious EN, line 46
   Delim off, line 1254
   New delims , line 1278
   New delims , line 1635
    in EQ, line 1635
   New delims ##, line 1991
   Delim off, line 1999
hostname%
```

We left out the `.EQ` before the `.EN` on line 46 to show you some sample output. This also reports on the delimiters. You can also use `checknr` with specific options to check specifically for a particular macro pair. For example, to run `checknr` to check that there is an `.EQ` for every `.EN`, type:

```
hostname% checknr -s -f -a.EQ.EN eqn.ug
46: Unmatched .EN
hostname%
```

Specify the macro pair you want to check for with the **–a** option and the six characters in the pair. The `–s` option ignores size changes and the `–f` option ignores font changes. See `checknr`(1) in the *SunOS Reference Manual* for more details.

Inline equations can only be so big because of an internal buffer in `troff`. If you get a message `word overflow`, you have exceeded this limit. If you print the equation as a displayed equation, that is, offset from the body of the text with `.EQ` and `.EN`, this message will usually go away. The message `line overflow` indicates you have exceeded an even bigger buffer. The only cure for this is to break the equation into two separate ones.

On a related topic, `eqn` does not break equations by itself; you must split long equations up across multiple lines by yourself, marking each by a separate `.EQ ... .EN` sequence. `eqn` does warn about equations that are too long to fit on one line.

7.22. Precedences and Keywords

If you don't use braces, `eqn` will do operations in the order shown in this list.

dyad vec under bar tilde hat dot dotdot
fwd back down up
fat roman italic bold size
sub sup sqrt over
from to

The operations that group to the left are:

over sqrt left right

All others group to the right. For example, in the expression

```
.EQ
a sup 2 over b
.EN
```

`sup` is defined to have a higher precedence than `over`, so this construction is parsed as $\frac{a^2}{b}$ instead of $a^{\frac{2}{b}}$. Naturally, you can always force a particular parsing by placing braces around expressions.

Digits, parentheses, brackets, punctuation marks, and the following mathematical words are converted to Roman font when encountered:

```
sin  cos  tan  sinh  cosh  tanh  arc
max  min  lim  log  ln  exp
Re  Im  and  if  for  det
```

The following character sequences are recognized and translated as shown.

Table 7-1 *Character Sequence Translation*

You Type	Translation
>=	≥
<=	≤
==	≡
!=	≠
+-	±
->	→
<-	←
<<	≪
>>	≫
inf	∞
partial	∂
prime	′
approx	≈
nothing	
cdot	·
times	×
del	∇
grad	∇
...	...
,....,	,....,
sum	Σ
int	∫
prod	Π
union	∪
inter	∩

To obtain Greek letters, simply spell them out in whatever case you want:

Table 7-2 *Greek Letters*

You Type	Translation	You Type	Translation
DELTA	Δ	iota	ι
GAMMA	Γ	kappa	κ
LAMBDA	Λ	lambda	λ
OMEGA	Ω	mu	μ
PHI	Φ	nu	ν
PI	Π	omega	ω
PSI	Ψ	omicron	ο
SIGMA	Σ	phi	φ
THETA	Θ	pi	π
UPSILON	Υ	psi	ψ
XI	Ξ	rho	ρ
alpha	α	sigma	σ
beta	β	tau	τ
chi	χ	theta	θ
delta	δ	upsilon	υ
epsilon	ε	xi	ξ
eta	η	zeta	ζ
gamma	γ		

The eqn keywords, except for characters with names, follow.

Table 7-3 eqn *Keywords*

above	lpile
back	mark
bar	matrix
bold	ndefine
ccol	over
col	pile
cpile	rcol
define	right
delim	roman
dot	rpile
dotdot	size
down	sqrt
dyad	sub
fat	sup
font	tdefine
from	tilde
fwd	to
gfont	under
gsize	up
hat	vec
italic	~, ^
lcol	{ }
left	"..."
lineup	

7.23. Several Examples

Here is the complete source for several examples and for the three display equations in the introduction to this chapter.

Square root

Input:

```
.EQ
x = {-b +- sqrt{b sup 2 - 4ac}} over 2a
.EN
```

Output:

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

Summation, Integral, and Other Large Operators

Input:

```
.EQ
lim from {x -> pi /2} ( tan~x) = inf
.EN
```

Output:

$$\lim_{x\to\pi/2}(\tan x)=\infty$$

Input:

```
.EQ
sum from i=0 to infinity x sub i = pi over 2
.EN
```

Output:

$$\sum_{i=0}^{\infty}x_i=\frac{\pi}{2}$$

Input:

```
.EQ
lim from {x-> pi /2} ( tan~x) sup{sin~2x}~=~1
.EN
```

Output:

$$\lim_{x\to\pi/2}(\tan x)^{\sin 2x}=1$$

Input:

```
.EQ
define emx "{e sup mx}"
define mab "{m sqrt ab}"
define sa "{sqrt a}"
define sb "{sqrt b}"
int dx over {a emx - be sup -mx}~=~
left { lpile {
     1 over {2 mab} ~log~
          {sa emx - sb}over{sa emx + sb}
above
 1 over mab~tanh sup -1 ( sa over sb emx )
above
 -1 over mab~coth sup -1 ( sa over sb emx )
}
.EN
```

Output:

$$\int \frac{dx}{ae^{mx}-be^{-mx}} = \begin{cases} \frac{1}{2m\sqrt{ab}} \log \frac{\sqrt{a}e^{mx}-\sqrt{b}}{\sqrt{a}e^{mx}+\sqrt{b}} \\ \frac{1}{m\sqrt{ab}} \tanh^{-1}(\frac{\sqrt{a}}{\sqrt{b}}e^{mx}) \\ \frac{-1}{m\sqrt{ab}} \coth^{-1}(\frac{\sqrt{a}}{\sqrt{b}}e^{mx}) \end{cases}$$

Quoted Text

Input:

```
.EQ
lim~ roman "sup" ~x sub n = 0
.EN
```

Ouput:

$$\limsup x_n = 0$$

Big Brackets

Input:

```
.EQ
left [ x+y over 2a right ]~=~1
.EN
```

Output:

$$\left[\frac{x+y}{2a}\right] = 1$$

Fractions

Input:

```
.EQ
a sub 0 + b sub 1 over
 {a sub 1 + b sub 2 over
  {a sub 2 + b sub 3 over
   {a sub 3 + ...}}}
.EN
```

Output:

$$a_0+\cfrac{b_1}{a_1+\cfrac{b_2}{a_2+\cfrac{b_3}{a_3+\cdots}}}$$

Input:

```
.EQ I
G(z)~mark =~ e sup { ln ~ G(z) }
~=~ exp left (
sum from k>=1 {S sub k z sup k} over k right )
~=~  prod from k>=1 e sup {S sub k z sup k /k}
.EN
```

Output:

$$G(z)=e^{\ln G(z)}=\exp\left(\sum_{k\geq 1}\frac{S_k z^k}{k}\right)=\prod_{k\geq 1}e^{S_k z^k/k}$$

Input:

```
.EQ I
lineup = left ( 1 + S sub 1 z +
{ S sub 1 sup 2 z sup 2 } over 2! + ... right )
left ( 1+ { S sub 2 z sup 2 } over 2
+ { S sub 2 sup 2 z sup 4 } over { 2 sup 2 cdot 2! }
+ ... right ) ...
.EN
```

Output:

$$=\left(1+S_1 z+\frac{S_1^2 z^2}{2!}+\cdots\right)\left(1+\frac{S_2 z^2}{2}+\frac{S_2^2 z^4}{2^2\cdot 2!}+\cdots\right)\cdots$$

Input:

```
.EQ I
lineup =  sum from m>=0 left (
sum from
pile { k sub 1 ,k sub 2 ,..., k sub m  >=0
above
k sub 1 +2k sub 2 + ... +mk sub m =m}
{ S sub 1 sup {k sub 1} } over {1 sup k sub 1 k sub 1 ! } ~
{ S sub 2 sup {k sub 2} } over {2 sup k sub 2 k sub 2 ! } ~
...
{ S sub m sup {k sub m} } over {m sup k sub m k sub m ! }
right ) z sup m
.EN
```

Output:

$$=\sum_{m\geq 0}\left(\sum_{\substack{k_1,k_2,\ldots,k_m\geq 0\\ k_1+2k_2+\cdots+mk_m=m}} \frac{S_1^{k_1}}{1^{k_1}k_1!}\,\frac{S_2^{k_2}}{2^{k_2}k_2!}\cdots\frac{S_m^{k_m}}{m^{k_m}k_m!}\right) z^m$$

Shorthand for In-line Equations

Input:

```
.EQ
delim ##
.EN
```

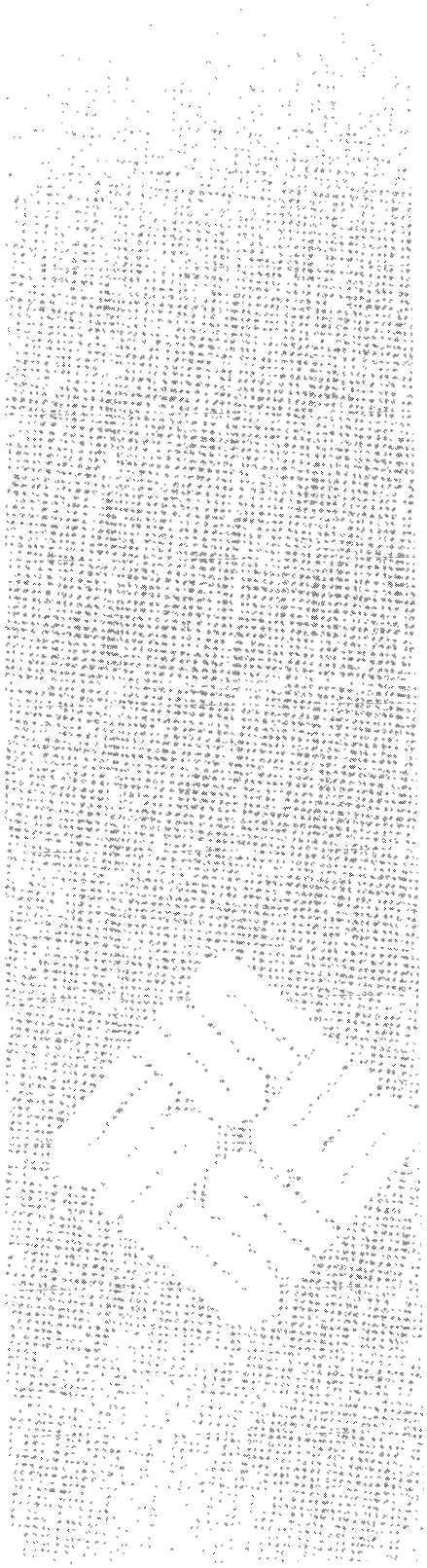
```
Let #x sub i#, #y# and #alpha# be positive
```

Output:

Let x_i, y and α be positive

8

Verification Tools



8

Verification Tools

8.1. `spell` — This command returns a list of misspelled words in a file. Because of the limited size of the on-line dictionary — less than 25,000 words — some words `spell` thinks are misspelled are in fact correct.

8.2. `checknr` — This program checks the syntax of `troff` files, in much the same way `lint` checks the syntax of C programs. People who try it often find it very helpful.

8.3. `soelim` — This program follows `.so` commands in `troff` files, incorporating the contents of these sourced files into the output. This program is helpful for searching groups of source files, and is also useful with preprocessors such as `refer`, `tbl`, and `eqn`, none of which follow source commands to fruition.

8.4. `deroff` — This command removes `troff` constructs from source files, and sends the results to standard output. Because some `troff` constructs necessarily contain text, some information may be lost from the output.

8.5. `fmt` — This command is a simplified formatter for use inside `vi` or `mail`. Devoid of hyphenation facilities, it does very little except fill text.

8.6. `col` — This command takes two-column text from `nroff` containing reverse line-feed escape sequences for the model 37 Teletype, and displays the two columns side-by-side, so they can be printed on a dumb lineprinter.

8.7. `colcrt` — This command is analogous to `col`, but was designed for CRT terminals, as it makes use of terminal capabilities when available.

8.8. `ul` — Also designed for CRTs, this command highlights underlined text using a terminal's underline mode, if available, and otherwise reverse video mode.

Index

Notes