Introduction to Text Processing

Editor's Note: Cross references in the text refer to chapters in the companion book, *UNIX: The Complete Reference, Second Edition,* by Rosen, Host, Klee, Farber, and Rosinski. The material in this chapter is largely historical and helps understand the roots of text processing in the UNIX environment. With the advent of newer word and text processing tools such as OpenOffice, TeX, and **groff**, some of which are discussed in this chapter, many of the original text processing tools that were part of early UNIX have been replaced with these platforms. In addition, a number of applications supporting text processing may not be commercially available today but may be running on legacy systems.

If you're a typical computer user, word processing and document preparation are among your most common tasks. Tools for preparing documents have been part of the UNIX System since its early days; the development of a text processing application to prepare patent requests was one of the things that led the management of Bell Laboratories to support the development of the UNIX System. The document preparation tools accompanying the UNIX System, known as the **troff** system, are flexible and powerful, although they are not as easy to use as word processing programs. They were once widely used for everything from basic tasks, such as producing business letters and writing memoranda, to complicated tasks, such as developing product documentation and typesetting professional articles and books. It is still worthwhile to know something about the **troff** system, since these tools may still be used in your organization. This information may also be helpful if you need to maintain old documents prepared using the **troff** system.

This chapter introduces the subject of text processing on UNIX. Here, you will learn about **troff** (pronounced "tee-roff"), the basic UNIX text-processing program, and related programs. You will learn about differences between text formatting systems and "What you see is what you get" (WYSIWYG) text processing systems. You will see how to use the ***mm*** *macros* (*memorandum macros*) to simplify using **troff** to prepare common types of documents such as letters. You will also learn about tools that check spelling, punctuation, and word usage, and pick up pointers on improving your writing. In particular, you will see how to check spelling by using **spell**, including how you can filter out customized lists of words, names, and acronyms from lists of supposedly misspelled words. You will learn about the *Writer's Workbench* and how it checks grammar, punctuation, diction, split infinitives, double words, sentence structure, and writing level.

This chapter also provides an introduction to TeX, a text formatting program developed by Donald Knuth, a well-known computer scientists at Stanford University. TeX is used

extensively for typesetting documents in the mathematical and physical sciences, as well as in computer science.

In addition to **troff** (and its associated programs) and TeX, several popular word processing packages are available for UNIX System computers; some of these are described at the end of this chapter. Although these word processing packages are usually easy to use, they are often less flexible than the standard UNIX System text processing tools.

The chapter "Advanced Text Processing" on this web site is devoted to additional UNIX text preparation tools, including those used to format tables, equations, pictures, and graphs. In these two chapters, you will learn about the wide range of UNIX System tools available for document preparation.

troff History

The ancestor of the **troff** program is a program called **runoff**, developed in 1964 at MIT. The first text formatting program for the UNIX System was **roff**, which was small and easy to use but could produce only relatively simple documents on a line printer. In 1973, Joe Ossanna of Bell Labs developed a more versatile and powerful text formatting program, called **nroff** (pronounced "en-roff"), short for "*n*ew *r*un*off*." Later in 1973, when a small typesetter was acquired by Bell Labs, Ossanna extended the capabilities of **nroff**, and the resulting program was called **troff**, which is short for "*t*ypesetter *r*un*off*." It is often forgotten in this day of desktop publishing that **troff** was the first electronic publishing program to exist for true typesetting. **troff** was originally designed to have output printed on a typesetter known as the C/A/T. To eliminate this dependency, Brian Kernighan revised **troff** so that it could send its output to other devices, including displays and printers. This revised version of **troff** is called *device-independent* ***troff*** and is sometimes known as **ditroff**. Today when most people refer to **troff,** they mean the **ditroff** program. The **ditroff** program is part of OpenSolaris.

Linux users will find that there is a GNU version of **troff**; it is called **groff**, which is widely available on the web. The GNU version includes all the capabilities of the **troff** system.

troff and the UNIX Philosophy

The **troff** system was designed and has evolved in line with the basic UNIX philosophy. That is, a series of tools and "little languages" have been developed to make text processing easier and to solve different types of text preparation problems. These tools include special packages of instructions called *macros* that make it easy to prepare particular types of documents such as letters and memoranda. The little languages developed as part of the **troff** system include *preprocessors* used to carry out special text formatting tasks such as building tables, formatting mathematical equations, drawing pictures, and creating graphs. Finally, there are **troff** *postprocessors,* which take **troff** output and prepare it for output on different types of devices such as PostScript printers.

These text preparation utilities are included with many different versions of the UNIX System. The utilities available for different variants of UNIX are often based on a package of software developed at AT&T Bell Laboratories called the Documenter's Workbench (DWB). The latest version of this software, DWB Release 3.4, was released in 1994 as an add-on software package available for computers running UNIX System V Release 4. The utilities in DWB 3.4 include the standard programs for text formatting, macro packages, preprocessors,

postprocessors, and other tools such as those used to produce indices. Another addition to the Documenter's Workbench was DWBX 3.4, an add-on package designed for use with the X Window System. Unfortunately, DWB Release 3.4 and DWBX 3.4 may be difficult to obtain.

The GNU project provides a public-domain version of the Documenter's Workbench, called **groff**, which is included as part of Linux. You can download **groff** from the GNU project site, at http://www.gnu.org/software/groff/.

troff Versus nroff

The basic programs used for text processing on the UNIX System are **troff** and **nroff**. You use **troff** when your output device is a typesetter, a laser printer, or a bitmapped display. You use **nroff** when your output device is a line printer or line-oriented display. **nroff** provides a subset of the capabilities of **troff** that work with line-oriented output devices. Because **nroff** commands form a subset of **troff** commands, **nroff** will be mentioned only when necessary. (In some versions of the **troff** system, **nroff** has been replaced with a "constant width" option for **troff**. On Linux, the equivalent of **troff** is called **groff**.)

The Text Formatting Process

To prepare a document using the **troff** (or on Linux, **groff**) you first create a file that includes both your text and formatting instructions. The formatting instructions are used to do such things as

- Center a line of text
- Skip a line
- Print text in a particular typeface such as italics or Helvetica
- Produce text in a specified point size ranging from very small to extremely large
- Print special symbols such as Greek letters, mathematical symbols, trademark symbols, and so on

You create your file containing text and formatting instructions using a text editor such as **vi**, discussed fully in Chapter 5 of the book. (You cannot use a word processing package to do this unless you filter your file to remove special control characters to obtain an ASCII file.) Then you run the **troff** program on this file; **troff** formats your text according to the **troff** codes contained in your file. You run postprocessing software on the output of **troff** to display the formatted document on your screen, or you pipe the output of **troff** to a printing command to print it.

This batch approach, first creating a document that includes formatting instructions and then using a program to format it, is different from the one-step, interactive approach used by WYSIWYG text processing systems that are commonly used on personal computers. When you are using a WYSIWYG system, your display shows at all times a close approximation of what will be printed. WYSIWYG systems available for UNIX computers will be described later in this chapter. When and why you should use text formatters will also be discussed.

Working with **troff** is similar to working with mark-up languages, such as SGML or HTML. This is no surprise, since **troff** served as the original inspiration when mark-up languages were first standardized in the 1980s.

Starting Out with troff

You use ***troff** commands* or *instructions* to tell **troff** how to format your text. To format your document, you first create a file that mixes your text with **troff** commands. There are two types of **troff** commands. One type of **troff** command is put on a line by itself, beginning with a dot. The other type of **troff** command occurs within a line of text and is called an *embedded command* because it is embedded in the line of text. This type of command begins with a backslash. For instance, the **troff** command

```
.ce
```

is used to center a line of text. It causes the *next* line of text to be centered.

The command

```
.sp 2
```

is used to space down two lines. This also illustrates the use of an *argument* to a command (here, **.sp** has the argument 2).

An example of an *embedded* **troff** command is **\fB**, which is used to change the font to boldface. For example,

```
\fBThis\fR formatting puts the word "This" in boldface.
```

Mix lines containing formatting instructions with text in a file in the following way:

```
.ce
EXAMPLE OF HOW TO USE TROFF COMMANDS
.sp 2
This example shows how troff commands, which are used for
different formatting tasks, are mixed with text into a single file.
This line is an \fIexample\fR of \s8how to use\s10
embedded troff commands.
This line of text follows a blank line in our input file.
```

This example contains two lines of **troff** commands, the first line **.ce**, which centers the next text line, and the third line **.sp 2**, which inserts two blank lines. There are then four lines of text. One of these has four embedded **troff** commands, **\fI**, **\fR**, **\s8**, and **\s10**. These commands change the font to italics, change the font to roman, change the point size to size 8, and change the point size back to size 10.

Several ways exist to produce formatted output from the file. To print output on the default printer connected to your system, run **troff** on the file (here the file is named *sample*) and pipe the output to **lp**, as follows:

```
$  troff sample | lp
```

To display the output on the screen, use **nroff** as follows:

```
$  nroff sample
```

troff puts words on a line until no room is left for another word. This is called *filling*. Also, **troff** puts some extra space between words so that the right margin is even. This is

called *right justification*. You also may have noticed that **troff** produced a blank line in the output from the blank line in input.

troff Commands Versus Macros

There are over 80 **troff** commands. You can use these commands to do almost any formatting task. You can even write "programs" (macros) that are combinations of these commands. Macros will be discussed in the chapter "Advanced Text Processing," also on this web site.

You have a lot of control when you work with **troff** commands. Each **troff** command deals with one small piece of the formatting task. However, using **troff** commands directly to format documents is difficult because you have to pay attention to many little details. When you use individual **troff** commands, you are providing the typesetter, printer, or display with detailed instructions about what it should do.

It would be tedious to format a long document using only individual **troff** commands. For instance, every time you start a new paragraph, you may need to use five different instructions. Fortunately, you can use macros, which group **troff** commands into a single higher-level instruction, to carry out a common task, such as starting a paragraph.

You can create your own macros to do the tasks you require. However, you also can use packages of macros that have already been developed. These macro packages contain instructions that can be used to carry out many common formatting tasks. Later in this chapter, you will learn how to use one of these packages, the **mm** macros (memorandum macros), to prepare common types of documents.

You can use **troff** commands and **mm** macros in the same document. Even when you are using the **mm** macros, you'll need to know some **troff** commands, because there are some common text formatting tasks that you cannot do with **mm** macros. A small group of frequently used **troff** commands will be introduced in this chapter. A broader set of **troff** commands will be discussed in the chapter "Advanced Text Processing."

Table 1 shows some **troff** commands that can be used to control where text is placed. There are **mm** macros that can be used for vertical spacing and starting new pages, but **.bp** and **.sp** are used frequently even when the **mm** macros are used.

Table 2 shows some commands that control the font and size of characters. There are versions that occur on separate lines beginning with dots, and there are inline versions beginning with a backslash. The first and third examples in the table are ones that would appear on separate lines, and the second and fourth examples in the table are ones that would appear inline.

Although there are **mm** macros that can do the same things, these **troff** commands are frequently used directly.

Command	Action
.in *n*	Indent all subsequent lines by *n* spaces
.br	Start new output line without adjusting current line
.ce *n*	Center next *n* input lines
.bp	Start new page
.sp *n*	Space vertically by *n*

Table 1 Some troff Commands for Controlling Text Placement

Command	Action
.ps *n*	Set point size to *n*
\S*n*	Set point size to *n*
.ft *f*	Switch to font *f*
\f*x*	Switch to font *x*

TABLE 2 Some troff Commands for Setting Point Sizes and Fonts

Including Comments

Sometimes you would like to include comments in your file that explain your formatting codes, especially when you have used many different **troff** commands or built macros. You want **troff** to ignore these comments, not treat them as text or as commands. You can put your comments in your file, because **troff** ignores a line that begins with a dot followed by a backslash and a double quotation mark. For example,

```
.\"  Put your comments here.
```

A line beginning with **\"** is considered a blank line of text by **troff**; including such a line will put a blank line in your document.

You can also include comments at the end of a line. Anything following the **\"** is ignored by **troff**. For instance,

```
.ce     \"This command centers the next line
```

The Memorandum Macros

The *memorandum macros* (or ***mm*** *macros*) package is a collection of instructions designed for formatting common types of documents. Each of these instructions is actually constructed of a sequence of individual **troff** commands. Macros generally have uppercase names and are used on separate lines beginning with dots. For example, the following is an **mm** macro for beginning a new paragraph:

```
.P
```

Memorandum macros are used to transform **troff** from a *procedural* language, where each command is in effect an instruction for the typesetting device, to a *descriptive* language, where each command represents a more human-oriented concept, such as a table, a heading, a paragraph, or a list.

The memorandum macro package is included with the Documenter's Workbench and is the most commonly used of all macro packages. In this chapter, you will learn how to use **mm** macros to format common types of documents. What follows is a discussion of how to format letters.

Formatting Letters

A wide variety of business letters can be formatted using a set of **mm** macros designed specifically for that purpose. There are **mm** instructions that format each element of

a typical business letter, such as the writer's address, the recipient's address, the salutation, and so on. These **mm** instructions determine the layout of the page, specify where each element of the letter is to be printed, and specify the size and fonts of the type used. Some **mm** macros take arguments or options. Sometimes any of several options can be used with an **mm** instruction to choose one of several possible formats. However, when you use **mm** instructions, you relinquish most of the control of the format of your document. In return, you can quickly and easily format common types of documents.

Arguments to **mm** macros must be enclosed in double quotation marks if the argument contains a blank space. If you do not use quotation marks, **troff** takes only the first word in the string as the argument.

Instructions for formatting a letter are displayed in the following *template.* This template shows the instructions used and the optional arguments they take. Each formatting instruction used in this template appears on a separate line beginning with a dot. The names of these instructions are easy to remember because they are abbreviations for what they do. For instance, the instruction **.WA** is used for *w*riter's *a*ddress, and **.WE** is used for *w*riter's *e*nd. The **.WA** and .**WE** pair follows the model of commands that start and end a special type of block. The template looks like this:

```
.WA "writer's name"
writer's address
.WE
.IA
recipient's name and address
.IE
.LO CN [confidentiality message]
.LO RN [reference subject]
.LO SA [salutation]
.LO AT [attention line]
.LO SJ [subject]
.LT [option]
Body option letter
.FC [formal closing]
.SG
```

The following example shows how a letter is formatted using this template:

```
.WA "Alice T. Tatum"          \"begin writer's address;
                              \"specify writer's name
Bulletin of Animal Parapsychology
111 Red Hill Road
Middletown, NJ 07748
United States of America
.WE                           \"end writer's address
.IA                           \"begin recipient's address
Zelda O. Quinn
23 Dorchester Place
London, England
.IE                           \"end recipient's address
.LO CN "private and confidential" \"confidentiality message
.LO RN "submitted article"    \"reference
```

```
.LO SA "Dear Ms. Quinn:"     \"salutation
.LT                          \"letter type
I am pleased to inform you that we have accepted your article
"Extrasensory Perception in Orangutans" for publication
in our journal. You will receive page proofs from us
in 2001.
.FC "Sincerely,"             \"formal closing
.SG                          \"signature line
```

Displaying and Printing Output

After you have formatted your letter, you can display it on your screen or print it. Suppose that your formatted letter is in the file *letter.* To display the letter on your screen, use the command

```
$ nroff -mm letter
```

The **–mm** option tells **nroff** to include the definitions of the **mm** macros when processing your document. You can also use the following equivalent command:

```
$ mm letter
```

To print your letter on the default printer, if it is a typesetter or laser printer, pipe the output of **troff** to the **lp** command. Use the command line

```
$ troff -mm letter | lp
```

or the equivalent command

```
$ mmt letter | lp
```

If your default printer is a line printer, use a similar command, with **nroff** instead of **troff**:

```
$ nroff -mm letter | lp
```

or the equivalent command

```
$ mm letter | lp
```

Although these commands will work with most output devices, you may occasionally obtain garbled output when you use certain output devices. This occurs when the default terminal type set for your system does not work for your terminal (or printer). If this happens, you need to supply the type of output device (CRT or printer). Do this by using the **–T***tty_type* option to these commands. For instance,

```
$ mm -Txterm letter
```

formats your letter for display on an X terminal.

The default device in DWB **troff** is PostScript, a common laser printer typesetting format.

The corresponding **groff** command for printing the file *letter* and sending the output to a printer (using the **–l** option of the **groff** command) is

```
$ groff -mm -l letter
```

This is the same as the following command line, where the output of **groff** is piped to the **lpr** command:

```
$ groff -mm letter | lpr
```

Macros for Letters

Table 3 summarizes some of the most important **mm** macros used for formatting a letter.

To specify several different components of your letter, use the **.LO** instruction with different options. You can use any combination of these options, ranging from none of them to all of them. For each one you want, use .**LO** with the appropriate option and the string you wish to use as the argument:

CN for the confidential line
RN for the reference line
AT for the attention line
SA for the salutation line
SJ for the subject line

If these instructions are used in your letter, they have to appear in the following order:

.WA
.WE
.IA
.IE
.LO (zero or more of these instructions with the appropriate arguments)
.LT

If the preceding instructions are not in the correct order, you will receive an error message when you try to print your document, or you will obtain output different from what you wanted.

Command	Description
.WA	Start of writer's address
.WE	End of writer's address
.IA	Start of recipient's address
.IE	End of recipient's address
.LO *option* [*argument*]	Letter *option*
.LT [*type*]	Business letter type (default is blocked)
.FC [*text*]	Formal closing
.SG	Signature

Table 3 mm Macros Used for Formatting Letters

Specify the type of letter you want by giving an argument to the **.LT** command. The following options are available:

- **BL (blocked)** This is the default, which starts all lines at the left margin except the date line, return address, and writer's identification, which begin at the center of their lines.
- **SB (semi-blocked)** This is the same as blocked, except the first line of each paragraph is indented five spaces.
- **FB (full-blocked)** This starts *all* lines at the left margin.
- **SP (simplified)** This is the same as full-blocked, except the salutation is replaced by an all-uppercase subject line followed by a blank line, the closing is omitted, and the writer's name is in all capital letters on one line.

The set of macros used to format a letter is a model of the sets of macros used to format other types of documents. That is, there are **mm** macros for each part of the letter, and you use the appropriate sets of macros for the elements of the letter in the correct order to format your letter. This same general approach is used for other common formatting tasks such as formatting business memoranda.

Formatting Memoranda

You have seen how to use **mm** macros to format letters. Now you will see how to use them to format business and professional memoranda. (As you might have guessed, the memorandum macros were originally designed for this very purpose.)

In general, a memorandum includes a block of information about the author, a title, a memorandum type, an optional abstract, the body of the memorandum, an optional formal closing, an optional signature line (or lines), an optional approval line (or lines), and an optional "copy to" list (or lists).

Here is a template to produce a memorandum:

```
.TL
title of memorandum
.AF
.AU "author's name"
.AT "title of author"
.AS
abstract
.AE
.MT [memorandum type]
body of memorandum
.FC [formal closing]
.SG
.AV name
.NS
list of recipients of memo
.NE
```

What follows is a description of how to use **troff** and the **mm** macros to format headings of different levels and lists of various kinds. An example will follow of how to use the preceding template.

Headings and Lists

Memoranda and technical documents are often organized into sections, subsections, subsections of the subsections, and so on. Each section or subsection covers a particular topic. Just looking at the sections and subsections produces an outline of the document. You can format headings into your documents easily using some built-in **mm** macros.

This is the instruction for producing a heading:

```
.H level [heading text]
```

The first argument to this instruction is the level number. The headings of the major sections of your document are level one headings, subsections of these receive level two headings, and so on. In all, seven levels of headings are allowed.

The second argument to this instruction is the optional heading text. You can use this to describe what the section or subsection is about. You use quotation marks around your heading so that the entire string is taken as the heading. If you do not use quotation marks, only the first word (the string of text preceding the first blank) will be used for the heading.

The sections and subsections of your document will be automatically numbered when you print out your document. This relieves you of having to keep track of section numbers manually. First- and second-level headings will be printed in italics (bold in earlier releases of DWB) followed by a blank line. Third- through seventh-level headings are printed in italics followed by two (horizontal) spaces. For instance, the instructions

```
.H 1 "UNIX System Text Preparation"
.H 2 "The troff System"
.H 3 "History"
.H 3 "Philosophy"
.H 2 "The mm Macros"
.H 3 "Writing Letters
.H 3 "Writing Memoranda"
```

produce the following:

1. UNIX System Text Preparation

1.1 The troff System

1.1.1 History

1.1.2 Philosophy

1.2 The mm Macros

1.2.1 Writing Letters

1.2.2 Writing Memoranda

Perhaps you can already guess how you could produce a synopsis of such a document just using tools you have learned already such as **grep** or **sed**.

The sample memorandum shown later in this chapter illustrates how to use levels of headings.

Unnumbered Headings

You can produce *unnumbered* headings using the **.HU** macro. For instance, the instruction

```
.HU "Wildlife of Madagascar"
```

produces the same heading that would be produced if .**H 1** were used, but without the numbers:

Wildlife of Madagascar

Be careful when mixing unnumbered headings and numbered headings, because some unnumbered headings change section numbers.

Formatting Lists

Formatting various kinds of lists is one of the most common tasks in the writing of memoranda, articles, and books. The **mm** macros provide versatile instructions for formatting many different types of lists. The commands you use to do this formatting follow a common model.

You begin a list with an instruction that specifies the list type, such as **.BL** for a *b*ulleted *l*ist or **.AL** for an *a*utomatically sequenced *l*ist. Next, you insert the individual list items, putting the instruction **.LI** (list item) before each item. Finally, you end the list with the .**LE** (list end) instruction.

For instance,

```
.BL
.LI
Huron
.LI
Ontario
.LI
Michigan
.LI
Erie
.LI
Superior
.LE
```

produces the *bulleted list*

- Huron
- Ontario
- Michigan
- Erie
- Superior

The following input,

```
.AL
.LI
```

```
Asia
.LI
North America
.LI
South America
.LI
Africa
.LI
Australia
.LI
Europe
.LI
Antarctica
.LE
```

produces a *numbered list*:

1. Asia
2. North America
3. South America
4. Africa
5. Australia
6. Europe
7. Antarctica

Available List Types

Table 4 describes the types of lists that can be produced by the **mm** macros by giving the instruction used to initialize the list.

Initial Instruction	How List Items Are Marked
.AL (or **.AL 1**)	Increasing Arabic numbers
.AL a	Letters in alphabetical order
.AL i	Lowercase Roman numerals
.AL I	Uppercase Roman numerals
.BL	Bullets
.DL	Dashes
.ML *mark*	User-specified marks
.VL	Variable; mark specified with each item

TABLE 4 Types of Lists

A Sample Memorandum

Following is a sample memorandum that uses the template previously introduced and the instructions for headings and building lists:

```
.TL                              \"memorandum title
Market Research Results
.AF "Monmouth Ice Cream Company"  \"alternate format; company name
.AU "Chip C. Chocolate"          \"author of memorandum
.AT "Director of Marketing"      \"author's title
.AS                              \"abstract start
This is a report on our market research on possible new
ice cream flavors for our ice cream parlors.
Our major findings indicate that
coconut ice cream would be successful in
Europe, but not in North America, and
peanut butter ice cream would be successful only in
North America.
.AE                              \"abstract end
.MT "Marketing Report"           \"memorandum type
.H 1 "Introduction"              \"first level heading
During May and June of this year an extensive market
survey was undertaken by the Monmouth Ice Cream
Company in Europe and North America.  We asked
one hundred people in each country where we
do business to sample two flavors of ice cream.
This memorandum states the results obtained and
gives recommendations to management.
.H 1 "The Results"               \"first level heading
We found that results varied by flavor and location.
.H 2 "Coconut Ice Cream"         \"second level heading
We found that coconut ice cream is a possible addition
to our line in some parts of the world.
.H 3 "North America"             \"third level heading
Only 23% of people who sampled coconut ice cream
in North America responded that they would buy this
flavor. Here are detailed results by countries:
.BL                              \"begin bullet list
.LI                              \"list item
Canada - 11%
.LI                              \"list item
Mexico - 44%
.LI                              \"list item
United States - 14%
.LE                              \"list end
.H 3 "Europe"                    \"third level heading
Our study found that 67% of people who sampled coconut
ice cream in Europe would buy this flavor.
.BL                              \"begin bullet list
.LI                              \"list item
France - 78%
.LI                              \"list item
Great Britain - 46%
.LI                              \"list item
```

```
Italy - 88%
.LI                              \"list item
Sweden - 62%
.LI                              \"list item
Germany - 61%
.LE                              \"list end
.H 2 "Peanut Butter Ice Cream"   \"second level heading
We found an extremely wide difference between
results for this flavor in North America and Europe.
.H 3 "North America"             \"third level heading
We found that 77% of people who tasted peanut
butter ice cream liked this flavor.
.H 3 "Europe"                    \"third level heading
Peanut butter ice cream was not widely accepted
in Europe. Only 2% of Europeans sampled liked this
flavor.
.H 1 "Recommendations"           \"first level heading
We make the following recommendations.
.AL                              \"begin numbered list
.LI                              \"list item
Coconut ice cream should not be introduced in North America,
except possibly in Mexico.
.LI                              \"list item
Coconut ice cream should be introduced in Europe, but
possibly not in Great Britain.
.LI                              \"list item
Peanut butter ice cream should be introduced throughout North America.
.LI                              \"list item
Peanut butter ice cream should definitely not be introduced in Europe.
.LE                              \"list end
.FC                              \"formal closing
.SG                              \"signature line
.AV "T. Frutti, President"       \"approval line
.NS                              \"begin "copy to" list
All members of New Flavors Department
.NE                              \"end "copy to" list
```

A Summary of Basic mm Macros

The basic **mm** macros for memoranda are summarized in Table 5. You must always include **.MT** (*m*emorandum *t*ype) and use the first seven of these commands in the order listed, if you use them at all.

Commonly Used mm Macros

Two sets of **mm** macros have been introduced for two specific tasks—formatting letters and memoranda. However, you will need a broader set of macros for general-purpose text formatting. There are **mm** macros that control text placement, the size and style (or font) of type, and page layout.

What follows is an introduction to the most commonly used **mm** macros for general tasks. These are used for such tasks as changing point size and vertical spacing, specifying

Command	Description
.TL	Title follows until next **mm** command
.AF [*company name*]	Alternate format with company name
.AU *name*	Author's name (up to 9 fields of information)
.AT *title*	Author's title
.AS	Start of abstract
.AE	End of abstract
.MT [*type*]	Memorandum type
.OK [*topic*]	Key-words
.PM [*type*]	Proprietary marking
.AV *name*	Approval line
.FC [*text*]	Formal closing
.SG [*name*]	Signature line
.NS [*type*]	Notation start; default is "Copy to"
.NE	Notation end

Table 5 mm Macros for Producing Memoranda

the font, turning on right justification (lining up right ends of lines), starting new paragraphs, inserting blank lines or skipping pages, and producing two-column output.

Because this is an introduction and a tutorial, all available options and arguments for commands will not be covered, nor will every **mm** macro. Instead, the most common uses of the most important **mm** macros will be covered. The use of these commands will be illustrated by formatting a résumé.

Controlling Point Sizes and Fonts

There are **mm** macros that are used to change point sizes, vertical space, and the font of the typefaces used.

Point Size and Vertical Spacing

The point size (the size of the type) and vertical spacing (space between the bases of two successive lines of text) of a document can be changed using the **.S** command. This is the form of the **.S** command:

```
.S size [spacing]
```

For example, the instruction

```
.S 9 11
```

changes the point size to size 9 and the vertical spacing to 11 points.

Command	Action
.B	Change font to bold
.I	Change font to italics
.R	Change font to Roman

TABLE 6 mm Macros for Changing Fonts

Fonts There are several **mm** instructions used to change fonts. Table 6 summarizes the most common of these.

If specified as in Table 6, all subsequent text is in the new font. If you want to change the font of only one word, it is easier to use the appropriate command with this word as an argument. The previous font will be reset immediately after the one word. For instance,

```
.B bananas
```

will put the word "bananas" in bold and then reset to the previous font.

Placing Text Several **mm** macros are commonly used to control the placement of text. These include commands used to start new paragraphs, to arrange for right justification, to skip lines or pages, and to produce two-column output.

Paragraphs The .**P** command is used to start a new paragraph. This command has the form

```
.P [type]
```

The argument specifies the type of paragraph, with 0 used for left justified; 1 for indented; and 2 for indented except after displays, lists, and headings.

Right Justification The command **.SA** is used to turn on or turn off right margin justification. The command

```
.SA n
```

turns off right margin justification if $n = 0$ and turns it on if $n = 1$. The default in **troff** is right margin justification; in **nroff** the default is no right margin justification. Justification is accomplished by widening the white space between words.

Skipping Vertical Spaces and Pages The macro **.SP** *n* produces *n* blank vertical spaces, and the command **.SK** *n* skips *n* pages.

Two-Column Output The **mm** macro **.2C** produces two-column output. The macro **.1C** returns the output to single-column output. In general, right margin justification should be turned off in two-column mode.

Summarizing Common mm Macros

The **mm** macros discussed are summarized in Table 7.

Macro	Action
.S *m n*	Set point size to *m* and vertical spacing to *n*
.B	Change font to bold
.I	Change font to italics
.R	Change font to Roman
.P	Start new paragraph
.SA 0	No right margin justification
.SA 1	Justify right margin
.SK *n*	Skip *n* pages
.SP *n*	Output *n* blank vertical spaces
.2C	Produce two-column output
.1C	Return to single-column output

Table 7 Some Commonly Used mm Macros

Accent Marks

You can use the **mm** macros to produce a variety of accent marks used by different languages. This is done using *escape sequences* that tell the formatter to place the appropriate mark on the preceding character. For instance, to produce a tilde over the letter *n* for the Spanish letter, you type

```
n\*~
```

Table 8 shows how to produce accent marks of various kinds using the **mm** macros.

Displays

Ordinarily, you want your text filled, justified, and positioned by **troff**. However, sometimes you may want to place a block of text on a page exactly the way you typed it, perhaps without any filling or justification of margins, and without breaking it across pages. Such a block is called a *display.*

Name	Input	Output
Acute accent	e*´	é
Grave accent	e*ì	è
Cedilla	c*	ç
Circumflex	o*^	ô
Tilde	n*~	ñ
Lowercase umlaut	u*:	ü
Uppercase umlaut	U*:	Ü

Table 8 Producing Accent Marks

The **.DS** macro is used for *static displays.* A static display appears in the same relative position in the text as it does in the input file. If the display is too large to fit on the current page, the rest of the current page is left blank and the display is printed on the next page.

The **.DF** macro is used for *floating displays*; if there is not enough space on the current page for a floating display, text *following* the **.DE** is used to fill the remainder of the page, and the display is placed at the top of the following page. Displays printed with the **.DF** command may be moved by **troff** relative to surrounding text, so that a floating display may appear after text that follows it (but not before text that precedes it).

The **.DE** command (for *display end*) marks the end of either a static or a floating display. Both the **.DS** and **.DF** instructions take three optional arguments used to set the format, fill mode, and indent of the display. The first argument is the format code; these are the arguments that can be used:

L	Do not indent (this is the default)
I	Indent the display (the default is 5 spaces)
C	Center each line individually
CB	Center the display as a block

The fill argument is next; it takes the value *N* for no-fill mode (this is the default) and *F* for fill mode. The third argument is the number of characters by which the line length for the display should be decreased, that is, a right indent. For instance, the command

```
.DS I N 10
```

is used for a display that is indented five spaces from the left margin, is printed in no-fill mode, and is indented ten spaces from the right margin. For instance, the following input text:

```
.DS I N 10
This is a sample display that illustrates
how to use the .DS command for a static display,
indenting it 5 spaces from the left, 10 spaces
from the right, in no-fill mode.
.DE
```

produces the following output:

```
This is a sample display that illustrates
how to use the .DS command for a static display,
indenting it 5 spaces from the left, 10 spaces
from the right, in no-fill mode.
```

A Sample Résumé

What follows is an illustration of how to use the different **mm** macros introduced to format a résumé. Here is the input file:

```
.ce                     \"center first text line that follows
.ps 15                  \"use point size 15
RESUME
```

```
.sp 3                   \"skip three lines
.S 11 13                \"change point size to 11 and vertical
                        \"spacing to 13
.DS                     \"start a static display
Otto E. Mattic
987 Navesink River Road
Red Bank, NJ 07701
(732) 555-5555
.DE                     \"end display
.S 10 12                \"change point size to 10 and vertical
                        \"spacing to 12
.sp 2                   \"skip two spaces
.P                      \"start paragraph
.B OBJECTIVE:           \"put word in bold and reset to roman
A position of authority at high pay in a UNIX System software company
.sp 2                   \"skip two spaces
.B EXPERIENCE           \"put word in bold and reset to roman
.sp                     \"skip a line
.BL                     \"start bullet list
.LI                     \"list item
July 1992 - present, President of Esoterix Software Corporation. Managed
production of UNIX System software for the entertainment industry.
.LI                     \"list item
April 1987 - June 1992, Member of Technical Staff, Taco Laboratories.
Wrote software to control production of fast foods.
.LE                     \"end list
.sp 2                   \"skip 2 lines
.B EDUCATION            \"put word in bold and reset to roman
.sp                     \"skip a line
.BL                     \"start bullet list
.LI                     \"list item
M.S. in Computer Science, University of Hawaii, June 1986.
.LI                     \"list item
B.S. in Mathematics, University of Alaska, June 1984.
.LE                     \"list end
.sp 2                   \"skip two lines
.B PUBLICATIONS         \"put word in bold and reset to roman
.sp                     \"skip a space
.AL                     \"start numbered list
.LI                     \"list item
"Artificial Intelligence in the Entertainment Industry,"
\fIJournal of Entertainment Technology,\fR
Volume 212 (1998), pages 110-221.
.LI                     \"list item
"Objective Oriented Programming for Fast Foods,"
\fITaco Technology Journal,\fR
Volume 2 (1997), pages 1-5.
.LE                     \"list end
```

Footnotes and References

The **mm** macros provide for formatting automatically numbered footnotes and references in documents. (This feature is missing in many desktop publishing packages.) To obtain

automatically numbered footnotes in your document, mark the spot where each footnote is to appear with the string ***F** and then follow this immediately with the block

```
.FS
footnote text
.FE
```

For example, enter the following:

```
A total of 4,300,000 UNIX System computers are
expected to be sold in 1999,\*F
.FS
According to a well-known market analyst
.FE
while only 3,700,000 were sold in 1998.
```

You will have an automatically numbered footnote placed at the bottom of the page containing this text, with a mark indicating the number for this footnote placed where the footnote occurs. You can also produce footnotes marked with any label you choose. Use the block

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

where the label is what you use to mark the footnote. For instance, the following input produces a footnote labeled with # at the bottom of the page containing this text:

```
This software program sold more than 1,000,000
copies in 1998 #
.FS #
800,000 for Windows and 200,000 for the UNIX System
.FE
and in 1999 it is expected to double its sales.
```

You can have reference lists automatically generated by marking the spots where you want to insert a reference number with the string ***(Rf** and using the block

```
.RS
reference text
.RE
```

Checking mm Macros

If you make formatting errors in a first version of your document, one way to find these errors is to print the document or to display it on your screen and check to see that nothing is amiss. Fortunately, there is an easier and quicker way to find certain types of errors made using **mm**. You can use the **checkdoc** command with the name of your file as the argument. Its output is a list of errors made in using the **mm** macros. For instance, **checkdoc** may find

a list not terminated with **.LE**, or macros used in the wrong order. Following is an example of a file containing **mm** instructions with several formatting errors:

```
$ cat letter
.LT "Dear Mary"
.WA "John Doe"
123 Main Street
Anywhere, USA
.WE
.IA
Mary Jones
321 First Street
Nowhere, USA
.IE

The latest information you requested is:
.DS
           1998 Revenue - $3,111,103,199.34
           1998 Profits - $0.37
.NE
.FC
.SG
```

The output obtained by **checkdoc** is

```
$ checkdoc letter
checkdoc diagnostics:
letter:
Line 1: Illegal argument for .LT
Line 1: .WA must precede .LT
Line 1: .WE must precede .LT
Line 1: .IA must precede .LT
Line 1: .IE must precede .LT
Line 2: Extra or out-of-sequence .WA
Line 5: Extra or out-of-sequence .WE
Line 6: Extra or out-of-sequence .IA
Line 10: Extra or out-of-sequence .IE
Line 17: .NE not preceded by .NS
Line 19: .SG not allowed within .DS/.DE pair
Missing .DE detected at EOF
19 lines done
```

The output of **checkdoc** is easily understood. Usually, it can be used to fix **mm** errors rapidly. For instance, in the preceding example, one way to fix the file *letter* is to correct the order of the letter macros, remove the incorrect argument for **.LT**, insert **.LO SA "Dear Mary"**, and change the incorrect **.NE** to **.DE**. (You should be able to do all these operations with your favorite editor by now.)

The **checkdoc** command was added in DWB 3.0; it is an enhancement to the **checkmm** command that was previously used. If you do not have DWB 3.0 or some other version of the **troff** system that includes **checkdoc**, the **checkmm** command may be available on your system. To see whether you have a Release 3 version of DWB, just type **dwbv**. The **dwbv** command, which does not exist on earlier releases, prints the DWB version number on later releases.

Printing Options

Command-line options can be used in your formatting commands to control output in various ways. For example, you can use such options to have the word "DRAFT" printed on the bottom of each page, to set the page width, and to print only specified pages.

Some mm and mmt Command Options

Following is a description of some useful options to the **mm** and **mmt** commands. You can identify your document as a draft or an official copy using the **–rC***n* option as follows:

–rC1 prints "OFFICIAL FILE COPY" on the bottom of each page.

–rC2 prints "DATE FILE COPY" on the bottom of each page.

–rC3 prints "DRAFT" on the bottom of each page, with single spacing of the document.

–rC4 prints "DRAFT" on the bottom of each page, with double spacing of the document and paragraph indents of ten spaces.

For instance, the following command will print your document formatted by **troff** with the word "DRAFT" placed on the bottom of each page:

```
$  mmt -rC3 section | lp
```

You can set the page width by using the **–rW***k* option to **mm** or **mmt**. For instance,

```
$  mm -rW7i section | lp
```

will print your document formatted by **nroff** with the length of each line equal to seven inches.

Some troff Command Line Options

Some useful options are available when you use the **troff** (or **nroff**) formatting command. Only the **troff** versions are given, but these commands work equally well with **nroff**.

Suppose that you have a document that is more than 200 pages long when printed. Your source file containing **troff** commands or **mm** macros is in the file *section1*. You make some small changes that will only alter page 47. You don't have to print out the entire document to see how this page has changed. Instead, you can print just page 47 using the command line

```
$ troff -mm -o47 section1  | lp
```

Similarly, you can print out a specified range of pages by using the **–o** option with a list of page numbers or ranges of page numbers. To illustrate this, use the following command line:

```
$ troff -mm -o-7,9,13-17,99- section1 | lp
```

to print out pages 1–7, 9, 13–17, and 99 to the end of the document.

The intent of the **–o** option is to save printer time and paper. You will not save processing time, however, because **troff** has to format all preceding pages as well, to assure that the requested pages are correct.

Sometimes you may not want a document you print to begin with page 1. For instance, you may be printing out the second chapter of a book, and the first chapter ends on page 46. To begin the pagination on page 47, use the command line

```
$ troff -mm -n47 chapter2 | lp
```

Other Macro Packages

In addition to the **mm** macros discussed in this chapter, several other macro packages are commonly used. The *ms* macro package, developed by Mike Lesk in 1974, was the first macro package to be used by a large number of people. It is still widely used and is available with many versions of the Documenter's Workbench. Another widely used macro package is *me*, written by Eric Allman at the University of California, Berkeley. This package is not part of DWB, but it is part of the SVR4 BSD Compatibility Package.

The *man* macro package is used to format UNIX System manual pages. It is available in DWB. There are also several macro packages used to format viewgraphs and slides. These are the *mview* and *mv* macro packages, also included in the Documenter's Workbench. You may find these, and perhaps other macro packages, on your system.

UNIX System Writing Aids: spell and Writer's Workbench

The **troff** system helps you format documents in an attractive and efficient way, but it does nothing to help you write well. Fortunately, the UNIX System provides tools to help improve your writing skills. You can check the spelling of words in your files using the **spell** program. You can check your punctuation, word usage, and writing level using the Writer's Workbench, a family of programs designed to improve writing.

spell

One advantage of using the UNIX System for text preparation is that you can use the **spell** command to check the spelling of words in any ASCII file. (This command only checks English language spelling.) Although spelling programs have become popular in the Windows world, the UNIX System **spell** program is the original algorithmic spelling program. A spelling program that is not algorithmic looks up every single word in a dictionary. **spell**, on the other hand, uses a database of roots, and rules for forming derivatives of these roots, such as plurals, to see whether each word is valid. As a result, it will find that "consider," "considers," "considering," and "consideration" are correctly spelled, while "considerz" is incorrectly spelled. **spell** handles lowercase and uppercase letters intelligently. It will accept "FISH," "Fish," and "fish," but not "fIsH."

The output of **spell** is a list of words that **spell** has determined are spelled incorrectly, as illustrated in the following example. First, the **cat** command will show you what is in the file *story*.

```
$ cat story
Once upn a time there was a system
administrater form America who walekd many
milse to see the great guru.
$ spell story
administrater
```

```
upn
milse
walekd
```

spell found four words it considers misspelled. Note that **spell** did not flag the word "form," since this is a correctly spelled word, even though this word should be "from" instead of "form." You would need a substantially more sophisticated program to pick up errors of this type. Once you use **spell** to find words spelled incorrectly, you can edit your file and correct them.

Because **spell** ignores lines starting with a period and embedded **troff** commands, you can use it to check the spelling of words in a file that you have formatted using the **troff** system.

Spelling According to British Rules

Spelling of words in English varies between the United States and the British Commonwealth. You can use **spell** to check whether words are spelled correctly according to British rules. To do this, you use the **–b** option. For instance, if you invoke **spell –b** on a file containing "theater" and "theatre," the output will contain "theater," but not "theatre." Here, **spell** listed "theater" as a misspelled word, but not "theatre," because "theatre" is the correct spelling in Great Britain. If you had invoked **spell** with no option, you would have had "theatre" and not "theater" in your list of misspelled words.

Using a Personal Spelling List

You may use words, names, or acronyms that **spell** does not consider correct. You can tell **spell** to ignore such words by giving **spell** a list of words that you consider correct.

To tell **spell** which words it should ignore, create a file containing these words, one on a line. Start with words beginning with uppercase letters in alphabetical order, followed by words beginning with lowercase letters in alphabetical order. An easy way to create this list is to create a file containing the words in any order, one on a line. Next, use **sort** (see Chapter 13 for a discussion of **sort** and related tools) to place these words in alphabetical order, one on a line, and redirect the standard output to a file. Then give the name of this file (containing the words, one on a line, in alphabetical order) preceded by a plus sign, as the first argument to **spell**.

Suppose that you use the word "workstation," the acronym "AFW," and the name "Tsai" in your files and do not want **spell** to tell you they are incorrect. Put these strings, one on a line, in a file named *temp*. Enter the command

```
$ sort temp > okspell
```

This produces a file named *okspell*:

```
$ cat okspell
AFW
Tsai
workstation
```

The following example shows how a file like *okspell* can be used to filter out words from the spelling list produced by **spell**. Suppose you have a file named *message*:

```
$ cat message
Our newe advanced functionality workstation (AFW) will be
developed by the laboratory headed by Howard Tsai.
```

When you run the **spell** command on this file, you obtain the following result:

```
$ spell message
AFW
Tsai
newe
workstation
```

This produced the misspelled word "newe," along with the correctly spelled name "Tsai," the correctly spelled word "workstation," and the acronym "AFW." To eliminate the words in your file, use the **spell** command with *+okspell* as an argument, as follows:

```
$ spell +okspell message
newe
```

The only output will be the misspelled word "newe."

Note that **spell** will not eliminate words from its output if they occur in the wrong order in your local spelling file, as you can discover by ordering these words incorrectly.

The Writer's Workbench

You can analyze your writing and check for spelling and grammatical errors by using the Writer's Workbench (WWB). WWB is a UNIX System V software package that is available separately. To have WWB analyze the contents of the file named *section1*, enter the command

```
$ wwb section1
```

Using this command runs a collection of programs that produce output covering many aspects of writing and grammar. This output includes possible spelling errors, sentences that **wwb** thinks may be punctuated incorrectly (and possible corrections), double words (such as "the the"), sentences with possible poor word choices or misused phrases (with suggested revisions), and split infinitives. You will also be given the Kincaid readability grade for your document, which indicates how many years of school someone needs to read your text and whether you have an appropriate distribution of sentence types, passives, and nominalizations (nouns formed from verbs).

To see the statistics calculated by **wwb**, you can **cat** the file *styl.tmp* produced by **wwb** when you ran it on your file. You will find readability grades for four different tests and information about your sentences. The information about your sentences includes their average length, the length of the longest sentences, distribution of sentence types, word usage statistics, and statistics on how you began your sentences.

You do not have to obtain a complete analysis of your writing if you are only interested in a particular aspect. You can run individual components of **wwb** separately to find out what you need. The component programs are **spellwwb**, which checks spellings; **punct**, which checks punctuation; **splitinf**, which checks for split infinitives; **double**, which checks for double words; **diction**, which checks word usage; **sexist**, which checks for sexist language; and **style**, which produces statistics on writing style.

Here is an example of the result obtained by running the command **double** on the file *section1*:

```
$ double section1
and and appears beginning line 202 section1
```

This output tells you that there is a double word, "and and," beginning on line 202 of the file *section1.* When you look at line 202 of this file, either it contains "and and" or it ends with "and" and line 203 begins with "and."

Running the command **style** on the file *section1* will give the following set of statistics:

```
$ style section1
style -mm -li wwb
readability grades:

                    (Kincaid) 10.4  (auto) 11.3  (Coleman-Liau) 11.0 \f(Flesch)
10.6 (56.9)
sentence info:

                    no. sent 59 no. wds 1200
                    av sent leng 20.3 av word leng 4.80
                    no. questions 2 no. imperatives 0
                    no. content wds 701  58.4%   av leng 6.12
                    short sent (<15) 32% (19) long sent (>30)  19% (11)
                    longest sent 56 wds at sent 50; shortest sent 5 wds at sent 24
sentence types:

                    simple  42% (25) complex  32% (19)
                    compound   8% (5) compound-complex  17% (10)
word usage:

                    verb types as % of total verbs
                    tobe  18% (26) aux  28% (40) inf  18% (26)
                    passives as % of non-inf verbs   5% (6)
                    types as % of total
                    prep 10.0% (120) conj 3.6% (43) adv 4.2% (50)
                    noun 28.8% (346) adj 15.2% (182) pron 7.0% (84)
                    nominalizations   2 % (21)
sentence beginnings:

                    subject opener: noun (13) pron (14) pos (0) adj (7) art (4) tot
64%
                    prep   5% (3) adv  17% (10)
                    verb   7% (4)  sub_conj   7% (4) conj   0% (0)
                    expletives   0% (0)
```

A final word on WWB: It offers *suggestions.* It is still up to you whether to heed them.

TeX

TeX is a text formatting program that has attracted a wide following. It was invented and developed by Donald Knuth, a well-known computer scientist from Stanford University, to typeset mathematics. Some versions of UNIX include a TeX package that includes the TeX and a variety of associated material. For example, Linux include TeTeX, which includes the TeX program, LaTeX (a collection of TeX macros defined for common desktop publishing tasks), several different TeX macro packages, and various printer and device printer programs, as well as support programs and documentation. You can also obtain public domain versions of TeX from the Comprehensive TeX Archive Network (CTAN), accessible

on the web at *http://ctan.tug.org*. A good way to learn about TeX is to access the TeX FAQs at *http://www.cogs.susx.ac.uk/cgi-bin/texfaq2html?introduction=yes*.

We will not explain how to use TeX to format documents here, since many easily available books discuss this topic. However, we will briefly describe the process used to print documents formatted with TeX. You may find this useful if someone sends you documents formatted with TeX, using any kind of computer, including a Macintosh, a Windows PC, or a UNIX computer.

A TeX Example

Suppose that the TeX formatted document is in a file named *paper.tex*. This is the first command you'll run to produce the document:

```
$ tex paper
```

This produces the output file *paper.dvi*, which contains code in a device-independent language used by TeX. A log file, *paper.log*, is created containing messages from the TeX program about the processing of the file *paper.tex*. (Notice that you give the **tex** command the filename without the filename extension *.tex*.) The next step is to produce a PostScript file from this intermediate file using this command:

```
$ dvips paper
```

This will process a file *paper.ps* in the PostScript page description language. You can then print out this file using the appropriate command to print out a PostScript file on your system. If you wish to display this file on your screen, you can use the **ghostview** command (which is available with many UNIX variants, including Linux, and is available as part of GNU).

By the way, if you use the X Window System, you can also use the program **xdvi** to view formatted TeX documents.

Translation Programs

Because different text formatters are available, people sometimes need to translate a document formatted with one formatter into the format used by a second formatter. Programs are available that translate documents formatted in **troff** to TeX, and that translate documents formatted in TeX to **troff**. For instance, the program **tr2tex** translates **troff** code to TeX, and the program **texi2roff** converts TeX files to **troff**. Be careful if you use translation programs, because such programs generally work for only a limited subset of commands of both formatters. Both these translation programs are available for the CTAN archives, accessible via the web site *http://ctan.tug.org*.

WYSIWYG Word Processors

This section describes some of the more popular word processing systems available on UNIX computers. It also lists some of their more important and interesting features, those beyond the standard text processing features that all WYSIWYG systems share. For a more comprehensive listing, see the *Open Systems Products Directory*, available on CD-ROM through UNIFORUM. Before buying a word processing package, make sure it supports your printer (or buy a new printer that is supported). Be especially careful of missing

features in these packages that may be critical to your document. For example, some of these packages lack page and section numbering, others cannot be used to produce tables, some cannot format footnotes, and so forth.

WordPerfect

Corel WordPerfect 8 for UNIX is a word processing application with many integrated features. It is available for a large number of hosts running different versions of UNIX, including SCO Open Server, HP-UX, AIX, and Solaris. WordPerfect 8 for UNIX has both a character-based interface and a graphical interface that runs on the X Window System. WordPerfect supports multiple columns, integrated comments and summary not intended to be printed, footnotes and endnotes, file management capabilities, a wide variety of formatting options, automatic indices and tables of contents, merging of sources, and sorting and searching. It can automatically create hyperlinks when it encounters the strings "www", "ftp", "http", or "mailto".

WordPerfect has features that allow tables to be built easily, and it supports a large number of spreadsheet functions that can be used with tables. It also offers spell checking and a thesaurus for searching for synonyms and antonyms, with both functions operating as you type. WordPerfect also supports macros for automatically executing a number of keystrokes. WordPerfect 8 provides a conversion filter for Microsoft Word 97 that can be used to import and export Microsoft Word 97 documents. The same file format is shared between Corel WordPerfect for Windows systems and Corel WordPerfect for UNIX, enabling cross-platform compatibility.

You can find out more about Corel WordPerfect for UNIX by visiting the Corel web site at *http://www.corel.com.*

Applix

Applix Words is a popular WYSIWYG word processor with a graphical interface that includes a rich set of desktop publishing capabilities. It is part of Applix Office, an integrated platform suite of desktop productivity tools, including word processing, spreadsheets, business graphics, graphics editing, data access, and e-mail. Applix Office was at one time available for a variety of UNIX variants, including AIX, HP-UX, Digital UNIX, Solaris, IRIX (for MIPS), UNIX System V Release 4, and SCO Open Server, as well as Windows and NT.

Applix Words includes a full set of standard word processing features, together with document support and integration with other Applix Office applications. Applix Words supports a wide variety of features, including mail-enabled compound documents; hypertext with HTML output; a forms editor; dictionaries, thesauri, and spell checkers; embedded equations and calculation support; tables and frames; and borders and shading. Applix Words also supports multimedia documents, including audio and video. You can learn more about Applix Words and Applix Office at *http://www.applix.com.*

StarOffice

StarOffice 5.0 is a cross-platform office productivity suite available from Star Division Corporation. Star Office runs on Solaris and Linux, as well as on Windows, Macintosh, and OS/2 platforms. It includes word processing with its StarWriter program, a spreadsheet program with its StarCalc program, graphic design with its StarDraw program, presentation support with its StarImpress program, database access with its StarBase program, scheduling with its StarSchedule program, and so on. StarOffice includes documents filters that provide

interoperability with Microsoft Office. StarOffice 5.0 is available free for individual noncommercial users. For more information about StarOffice, consult the Star Division web page at *http://www.stardivision/com.*

FrameMaker

FrameMaker 5.5.6 is a desktop publishing system available for Solaris, IRIX, and AIX from Adobe. FrameMaker offers a wide array of features. It provides all the features of a standard word processor, a spelling checker with an optional use of dictionaries in many different languages, and a punctuation checker. A rich set of page layout capabilities are supported by FrameMaker, such as the ability to produce pages with customized sizes, to automatically update various types of graphics elements, to mix portrait and landscape modes in the same document, and so on. FrameMaker has graphics capabilities, similar to those found in graphics packages, for creating drawings. FrameMaker also includes optional text filters that are used to integrate text formatted in other systems, including the **troff** system and several popular word processors, such as Microsoft Word, into FrameMaker documents. Graphics filters are included that are used to integrate graphic images generated by CAD programs or Macintosh MacDraw into FrameMaker documents. FrameMaker has a WYSIWYG math processor that is used to enter, format, simplify, and solve mathematical equations.

FrameMaker also provides tools for building large documents such as books. For instance, it includes tools for automatically generating indices, customizing pagination, adding running headers and footers, creating footnotes, and maintaining cross-references within and between multiple documents. FrameMaker can be used to print documents on PostScript printers or to produce PostScript files that can be sent to outside phototypesetters.

FrameMaker has the capability of having multiple windows open simultaneously. Text can be exchanged between windows and applications. A set of file management tools provides for automatic saving, cancellation of changes, a browser system, and mail capabilities. Finally, FrameMaker provides interfaces that can be used to import data from other software applications, such as spreadsheets and database packages.

Framemaker also supports Asian language publishing by including support for double-byte characters. Japanese, Chinese, and Korean characters can be imported into FrameMaker documents when the operating system supports multibyte characters and the appropriate fonts are supported.

You can learn more about FrameMaker at the web site *http://www.adobe.com/prodindex/framemaker/main.html* and from the newsgroup *comp.text.frame.*

Microsoft Word

Although Microsoft does not generally port its applications to UNIX, an earlier version of Microsoft Word will run on UNIX System V Release 4, based on its ability to run on Xenix. Microsoft Word is file-compatible with the corresponding DOS versions of Microsoft Word. In addition, it supports a wide range of editing and formatting features, including multiple columns, footnotes, hidden text, and automatic paragraph numbering. This word processor permits editing of two windows simultaneously. Automatic sorting is also supported. Microsoft Word prepares automatic outlines of documents, tables of contents, and indices, and includes style sheets for standardized documents. Microsoft Word performs basic mathematical functions.

Text Formatting Versus WYSIWYG Systems

After learning about text formatters and WYSIWYG systems, you may want to know when and why text formatters are preferable. You'll want to know which text processing program or programs you may want to use.

WYSIWYG systems have advantages for some common word processing tasks, whereas text formatting systems, such as the **troff** system and TeX, are better suited to others. For instance, you can use the **troff** system or TeX to create documents with any page layout you want, whereas on WYSIWYG systems it is either impossible or extremely difficult to produce documents different from those the system was designed to produce. When you write a document using a text formatting system, you first concentrate on the contents of the document and then customize the appearance by changing your formatting commands. You do not have this flexibility with WYSIWYG systems. (Nowadays, since many more people use TeX than the **troff** system, TeX is usually a better choice as a text formatting system. However, this may not be the case if you are in an environment where **troff** is still widely used.)

Text formatters have other advantages over WYSIWYG systems. For instance, WYSIWYG programs use a lot of processing resources, because they maintain the appearance of a document as it is being prepared. Making a small change such as adding or deleting a word can require extensive reformatting of the entire document. The result is that using WYSIWYG programs can slow down systems, especially multiuser systems (although on most modern workstations this is not an issue). Another advantage of text formatters over WYSIWYG systems is that you can use any text editor, such as **vi**, or any other tool, to change the format of your document, because you are working with an ASCII file. For instance, you can use **vi** to change the font used for the word "UNIX" from italics to boldface by making a global substitution, or run **spell**, on a **troff** document. You can't do this with documents prepared with WYSIWYG systems, because they produce files that contain non-ASCII characters. You can also transport your documents to any other system, since no special file format is needed.

You may find it preferable to use the troff system or TeX instead of a WYSIWYG system in the development of large documentation projects, although this can be done using desktop publishing systems, such as Framemaker, which tend to be somewhat expensive. The **troff** system and TeX are used extensively within companies and universities to coordinate and produce documentation for computer systems. The production of these large documentation projects is made easier by the use of UNIX System tools, such as the **make** program, originally designed to develop large software projects (see Chapter 29 for more information on this).

In practice, many people preparing documents compromise by using a windowed terminal, editing the **troff** or TeX source in one window and displaying the output in another. Although not WYSIWYG, this arrangement does provide immediate feedback of your changes.

UNIX System Text Processing Tools for Windows

Many people use the UNIX System on their computer at work and have a Windows PC at home. Other people may use both the UNIX System and Windows at work. If you want to use the same text formatter on both UNIX computers and Windows PCs, as well as on Macintosh computers from Apple, note that you can use TeX in all environments.

If you are a **troff** user and need to use **troff** on UNIX systems and on Windows machines, you can employ Eroff, a product of the Elan Computer Group. Eroff is available for Windows systems (running in DOS mode), as well as many different UNIX variants. Eroff, which is based on the **troff** system, supports most of the features provided by the **troff** system and its preprocessors and macro packages. It also provides additional features for the inclusion of graphics and for previewing output on a CRT screen. You can find out more about Eroff by consulting *http://www.elan.addr.com/products/eroff.htm.*

Summary

This chapter was devoted to getting you started with text preparation in the UNIX System. The **troff** system was introduced and its philosophy described. You learned how to use some basic **troff** commands and the memorandum macros to carry out some common text formatting tasks. You learned about tools for checking spelling and grammar. The TeX system was introduced and described. Finally, a survey was given of the available UNIX System software for text processing, including text formatters, WYSIWYG word processors, and desktop publishing systems.

In "Advanced Text Processing," a variety of more advanced topics in text preparation on the UNIX System will be covered. Tools will be introduced for specialized formatting tasks such as producing tables, formatting mathematical equations, and drawing pictures. You will see how to customize page layout and the overall appearance of your documents. You will also learn about some of the features available for inserting images generated with the PostScript page description language into documents prepared with the **troff** system. Also, you will learn how to use some of the tools available for using **troff** on the X Window System.

How to Find Out More

Readers who want detailed information about the topics covered in this chapter can consult the following books devoted to text preparation with the **troff** system. Note that most of them are out of print and might be available in used book stores or via web services such as eBay:

AT&T. *Documenter's Workbench Release 3.4,* DWB Documentation. Murray Hill, NJ: AT&T Bell Laboratories, 1993.

Christian, Kaare. *The UNIX Text Processing System.* New York: Wiley, 1987 (out of print).

Dougherty, D., and T. O'Reilly. *UNIX Text Processing.* Hasbrouck Heights, NJ: Hayden Book Company, 1988 (out of print).

Emerson, S.L., and K. Paulsell. *troff Typesetting for UNIX Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1987 (out of print).

Gehani, N. *Document Formatting and Typesetting on the UNIX System.* 2nd ed. Summit, NJ: Silicon Press, 1987.

Gehani, N., and S. Lally. *Document Formatting and Typesetting on the UNIX System,* Volume II. Summit, NJ: Silicon Press, 1988.

Kernighan, Brian. "The UNIX System Document Preparation Tools: A Retrospective" *AT&T Technical Journal,* Volume 68, Number 4 (July/August 1989).

Roddy, K.P. *UNIX nroff/troff, a User's Guide.* Englewood Cliffs, NJ: Prentice-Hall, 1987 (out of print).

Srinivasan, B., Unix Document Processing and Typesetting. World Scientific, 1993.

Readers who want information about TeX and LaTeX may want to consult the following books:

Goossens, M., F. Mittelbach (contributor), and Alexander Samarin (contributor). *The LateX Companion.* Reading, Massachusetts: Addison-Wesley, 1994.

Knuth, D.E. *The TeXbook: A Complete Guide to Computer Typesetting with TeX.* Reading, Massachusetts: Addison-Wesley, 1988.

Krantz, S.G., and S.A. Sawyer. *A TeX Primer for Scientists.* CRC Press: Boca Raton, 1995.