The Software Toolworks
Walt Bilofsky, Prop.

14478 GLORIETTA DRIVE
SHERMAN OAKS, CALIFORNIA 91423

TELEPHONE
(213) 986-4885

TEXT (Version 4.0)
Text Formatter
28 March 1982

Dr. James J. Gillogly
2520 S. Chard Ave.
Topanga CA 90290
(213) 455-1407

TEXT USER'S MANUAL
by Marrietta Gillogly

TABLE OF CONTENTS

USER'S MANUAL
TEXT (Version 4.0)
Text Formatter

1. INTRODUCTION

WHAT IS TEXT?

TEXT is a program for formatting documents. It will automatically divide your document into pages with space at the top and bottom, even margins, specified headings, and page numbers. It will underline, embolden, center, hyphenate, indent, and overstrike specified words or phrases. It will even create a table of contents, help create an index, and generate form letters.

TEXT is ideal for formatting letters, papers, articles, outlines, books, stories, invoices, proposals, and documentation such as this. It is no doubt useful for numerous other applications as well. Since versatility has been programmed into it, its use can be tremendously enhanced by the ingenuity of the person using it. You may control most of its functions, such as specifying the number of lines you want a page to be (11" paper is 66 lines, but you could specify 84 lines for legal size 14" paper instead), how large you want the spaces at the top and bottom of the page, or what number you want the page numbering to begin with.

HOW DOES IT WORK?

Using TEXT to format a document requires two steps.

1. TEXT Formatting Instructions

The first step is to put TEXT instructions into your document file. You must tell TEXT how you want to format this document: which words you want underlined, which titles centered, where you want the page numbers to appear, etc. This is accomplished by putting TEXT instructions (described in Section 2 and summarized in Section 3 of this manual) into your document file using an editor such as PIE, MINCE, EDIT, ED, or EDIT19.

For instance, to center a title, you would insert a line with the centering instruction, ".ce" preceding the title line.

Figure 1 is an example of a document file with TEXT instructions.

2. The TEXT command

Once you have all the TEXT instructions you want in your document file, you are ready to run TEXT. When you run the TEXT program, TEXT will process your document file and execute the TEXT instructions as it encounters them in the file, formatting your document as you have specified. Figure 2 shows the formatted document obtained by running TEXT on Figure 1.

To run TEXT you simply give the text command and the name of the file to be formatted. For instance, to format the sample file sample.fle included on your distribution disk, type at the operating system prompt (A> for CP/M or > for HDOS):

```
>text sample.fle
```

The formatted output will appear on your screen.

You may have the formatted output from TEXT sent to a file or your printer instead of the screen simply by giving the name of the file or printer device as a part of the command:

```
>text sample.fle lp:          (for HDOS)
A>text sample.fle lst:        (for CP/M)
```

will print your formatted document on your line printer, or

```
>text sample.fle sample.txt
```

will store the formatted document on your disk, in a file called sample.txt, where you may check it to see if it is formatted just as you want.

Additional functions available from the command line are described in Section 4.

Figure 1: Input file.

```
.ce 2
.bd
TEXT (Version 4.0)
Text Formatter

.pp
TEXT is a program for formatting documents.
It will automatically divide your document into pages
with space at the top and bottom,
even margins, specified headings, and page numbers.
It will embolden, under\-line, center, hyphenate, indent,
and overstrike speci\-fied words or phrases.  It will even
create a table of contents and generate form letters.
```

Figure 2: Result of running TEXT on Figure 1.

TEXT (Version 4.0)
Text Formatter

TEXT is a program for formatting documents. It will automatically divide your document into pages with space at the top and bottom, even margins, specified headings, and page numbers. It will embolden, underline, center, hyphenate, indent, and overstrike specified words or phrases. It will even create a table of contents and generate form letters.

HOW TO USE THIS MANUAL

This manual is divided into eight sections as follows

1. Introduction: What is TEXT and How Does It Work
2. A TEXT Tutorial
3. Summary of TEXT Formatting Instructions
4. The TEXT Command
5. Special Uses: Form Letters, Table of Contents & Index, Footnotes
6. Trouble-shooting
7. Changing the Defaults: Patches
8. Index

The bulk of the manual is Section 2: A TEXT Tutorial. Every TEXT formatting instruction and option is presented and explained here with examples and exercises to help you gain a working knowledge of formatting with TEXT in a fairly short time. This section is essential reading for those who have not done word processing on a computer before. Even if you are already familiar with other types of formatting programs, you should at least look over this section to acquaint yourself with the particular instructions and capabilities available with TEXT.

Sections 3 and 4 are short summaries of the material presented in the tutorial section. This is provided as Ready Reference material for quick look-up and review once you have mastered the basics of TEXT.

Section 5 includes hints on how to use TEXT for special applications such as form letters and end-of-chapter footnotes.

Section 6 is an attempt to provide helpful suggestions when things go wrong for seemingly unknown reasons.

Section 7 will explain how you may change the program's defaults for your individual situation. It includes instructions on how to use the file text.pch supplied on your distribution disk.

The final section is the index, which will keep all the explanations and helpful hints at your fingertips, cross-referenced by instruction name, function, and usage.

****

TEXT is based on and greatly extended from format by Kernighan and Plauger. For additional information, see the reference:

Kernighan, B.W. and Plauger, P.J.; Software Tools; Addison-Wesley Publishing Company, Reading, MA; pp. 219-250.

2. A TEXT TUTORIAL

WHERE DO I START?

Before you do anything else, place a write protect label on your TEXT distribution disk and copy all the files from it onto another disk. Store your distribution disk in a safe place and use your other disk for text formatting.

Your TEXT disk should include the following files:

for HDOS disks:	for CP/M disks:
text.abs	text.com
text.pch	text.pch
sample.fle	sample.fle
sample2.fle	sample2.fle
sample.in	sample.in
sample.out	sample.out

If you run TEXT on a text file which has no TEXT instructions in it, TEXT will divide it up into pages of 66 lines each, including 5 blank lines at the top and 5 blank lines at the bottom. The left margin will be set at 1 and the right margin at 60, with all lines right justified (i.e., spaces will be added between words as necessary to make each line exactly 60 characters long). A blank or indented line will be assumed to begin a new paragraph so that the line preceding it will not be justified. Indented lines will be indented. Nothing will be underlined, centered, hyphenated, or emboldened.

The sample file sample.fle on your distribution disk is a text file that contains no TEXT instructions. Type the file on your screen and then run TEXT on it (type "text sample.fle" after your operating system prompt). Notice how TEXT has formatted the sample file. (You can use CTRL-s to stop the text from scrolling off your screen and CTRL-q to allow it to start scrolling again.)

To change this format we will need to add TEXT instructions to sample.fle. There are two types of TEXT instructions, "in-line" and "separate line" instructions. We will be working with the second type first. Each "separate line" TEXT instruction is placed on a line by itself, and must begin at the left edge of the line. It consists of a two-character name preceded by a period (.) and often followed by an argument which defines or modifies the instruction in some way. For instance, the right margin instruction is ".rm" followed by the character position at which the right margin is to be set (the argument).

RIGHT MARGIN

To change the right margin to 79 in our sample.fle we need to insert the line

```
.rm 79
```

at the beginning of the file. (Using your favorite editor, do that now.) Now when we run TEXT, (type "text sample.fle" after your operating system prompt) the justified lines will fill the screen. (If your screen is not 80 characters wide, substitute your screen width minus one for 79 in the right margin instruction above.)

LEFT MARGIN

To change the left margin, use the indenting instruction followed by the place where the left margin is to be set. For example,

```
10
```

will set the left margin at character position 10. Add this instruction to the beginning of the sample.fle, either before or after the ".rm" instruction, and then run TEXT again. Note that the lines which were indented 5 spaces from the left edge in the file are now indented 5 spaces from the left margin on output.

PAGE LENGTH

To change the page length, use the ".pl" instruction followed by the number of lines long each page is to be, including the 5 blank lines at the top (the header) and 5 blank lines at the bottom (the footer) of each page. The instruction

```
33
```

will give you pages that are 33 lines long. Add this instruction to the top of our sample file. So long as the instruction comes before the first line of text it does not matter in what order it is given in relation to the ".in" and ".rm" instructions.

Run TEXT on the sample.fle. Each page will be separated from the next by a block of 10 blank lines representing the footer of one page plus the header of the next page.

PAGE HEADER SPACE

To change the amount of space that is left blank at the top of each page, we will need to use the header instructions, ".h1" and ".h2". These are initially set at 3 and 2 respectively, to give us a total of 5 blank lines at the top of each page (there are two instructions instead of just one, to allow you to specify a page header with space above and below it before the body of the text). To cut this space down to one blank line we will add the following instructions to our sample.fle:

either

```
.h1 1
.h2 0
```

or

```
.h1 0
.h2 1
```

Either set of instructions will give us one blank line at the top of the page, since we have not yet specified a page header. We will go into this more later.

PAGE FOOTER SPACE

The footer space at the bottom of the page works the same way, with the instructions, ".f1" and ".f2". Let's change the footer space from the default 5 blank lines to only two blank lines. Since we do not have a page footer that we want to print at the bottom of each page, we can accomplish this in any one of three different ways. We'll use:

```
.f1 1
.f2 1
```

although we could have used

```
.f1 2
.f2 0
```

or

```
.f1 0
.f2 2
```

to obtain the same result.

Add instructions to the beginning of the sample.fle to get one blank line at the top and two blank lines at the bottom of each page.

JUSTIFICATION

As it stands now, TEXT will add enough spaces between words to make each line exactly the same length. This is known as justifying the text. If you would rather have only one space between each word and the line lengths only approximately even (known as a ragged right margin) you may use the ".nj" instruction to tell TEXT not to justify.

```
.nj
```

In our sample.fle, place this instruction after the second paragraph -- on a line by itself as with the other TEXT instructions. Now give the TEXT command (text sample.fle) and note the difference between the justified and non-justified text, as well as the new spacing at the top and bottom of each page.

When we want to return to justified text, we will use the instruction:

```
.ju
```

which will again add spaces between words to make the lines exactly the same length. TEXT will add spaces between words beginning at the right on the first line, beginning at the left on the second line, and alternating thereafter, so as to avoid a "thinning-out" at just one side.

FILL & NO FILL

There will be times when you will want TEXT to leave your lines just as you typed them, without altering their lengths at all. You will not want them justified; you will not even want them filled to similar lengths. The inside address of a letter, for instance, should be printed exactly, line-for-line, as it appears in your input file. So should tables, most examples, and poetry quotations.

To tell TEXT to stop filling lines, we use the no fill ".nf" instruction.

In our sample.fle, there is a list of the sections in this manual:

> This manual is divided into eight sections as follows:
>
> 1. Introduction: What Is TEXT and How Does It Work
> 2. A TEXT Tutorial
> 3. Text Formatting Instructions
> 4. The TEXT Command
> 5. Special Uses: Form Letters, Table of Contents & Index
> 6. Trouble-shooting
> 7. Changing the Defaults: Patches
> 8. Index

When these lines are filled, they are all run together. They will be much easier to read if we instruct TEXT to output them just as they appear in the file. Find this list near the middle of the sample.fle and insert the "no-fill" command on its own line directly above the first section listing:

```
.nf
```

When you want TEXT to resume filling (either for justified or ragged right margins, depending on whether a ".nj" or ".ju" was the most recent justification instruction) use ".fi" (fill). Just to see what happens, let's resume filling with the second paragraph after our list of sections. In the sample.fle insert the fill instruction:

```
.fi
```

Run TEXT (text sample.fle) and note that the list now appears with each section listed on a separate line, just as it is in the input file. However, the following paragraph, before the ".fi" instruction takes effect, is VERY ragged. Go back into the file now and move the fill instruction so that it takes effect immediately after the last item on the list.

INDENTING

To set our list off even more let's indent it 12 spaces to the right. Since we set our left margin to 10 at the beginning of the file with the command

```
.in 10
```

and have not touched it since, we know that by giving the instruction

.in 22

we would get our list indented 12 spaces from the left margin. However, TEXT makes it even easier by allowing you simply to state how much you want to move your left margin in or out, rather than having to remember where your indentation is currently set. To move our list 12 spaces to the right, we simply insert the instruction

.in +12

and when we want to return to our previous indentation, we instruct TEXT with

.in -12

which will move the left margin 12 spaces to the left.

The right margin instruction, ".rm" also accepts relative numbers as arguments, so that we could pull the right margin in 12 spaces from the right with the instruction

.rm -12

and return it to our original setting with

.rm +12

A positive number (+) will move the indentation or right margin setting to the right. A negative number (-) will move it to the left.

In our sample.fle, insert the instructions to indent the list of sections 12 spaces to the right and to pull the right margin in 12 spaces to the left. (These may be placed either before or after the no-fill ".nf" instruction, but must each be on a line by itself with the period "." in column 1.) After the list and its following paragraph, return both margins to their original settings.

You should have inserted the instructions ".in +12" and ".rm -12" before the list, and the instructions ".in -12" and ".rm +12" on lines after the paragraph which follows the list.

Run TEXT now to see how it worked. Did the margins return to their previous settings? Did you notice that the new right margin setting did not affect the list? It turns out that TEXT in no-fill mode ignores the right margin. If the line you input is longer than your output line, TEXT will print that line into the right margin. It will print it EXACTLY as it appears on input, regardless of the right margin setting or the line length. When you are displaying output on your screen, lines that are longer than your screen width will be displayed on two lines with the second line beginning at the left edge regardless of your left margin setting.

If you want to have TEXT split long lines at the right margin, you must remove the no-fill ".nf" instruction, and instead use a break ".br" instruction after each list entry. This instruction tells TEXT to stop filling this line and begin a new one. Your input file would look like this:

```
1. Introduction: What Is TEXT and How Does It Work
.br
2. A TEXT Tutorial
.br
...
5. Special Uses: Form Letters, Table of Contents & Index
.br
...etc...
```

and TEXT would format it like this:

1. Introduction: What Is TEXT and How Does It Work
2. A TEXT Tutorial
...
5. Special Uses: Form Letters, Table of Contents & Index
...etc...

Try this now by removing the no-fill (.nf) instruction at the beginning of the list of manual sections, and inserting a break (.br) instruction between each list entry. When you run TEXT, pay particular attention to the way the list is formatted. How could we make it more attractive?

HANGING INDENTS

If we are going to have some of our list entries on two lines, it would look nicer to indent each second line to correspond with the first word on the first line rather than with the number. We can do this with TEXT by using the hanging indent ".hi" instruction to produce, for instance,

1. Introduction: What Is TEXT and How Does It Work

We need to include in the instruction, the characters to be "hung" (the hanging indent) and how far to the left of the current indent they are to "hang". In our current example, for instance, we want the "1. " to hang to the left of the block of filled text, three spaces into the left margin. We specify this with ".hi 1. -3" where the first argument to the ".hi" instruction is the word, phrase or characters to be "hung" (1.) and the second argument is the number of spaces (3) the first argument is to begin to the left (-) of the current indentation. The first entry in our list will look like this:

```
.hi 1. -3
Introduction: What Is TEXT and How Does It Work
```

If we omit the minus sign (-) in the second argument, TEXT will use the number as the absolute character position at which to begin the hanging indent. For instance,

```
.hi 1. 3
Introduction: What Is TEXT and How Does It Work
```

would format as

1. Introduction: What Is TEXT and How Does It Work

with the hanging indent (1.) begining at character position 3 and the text block beginning at the current indent level.

If the hanging indent overlaps the left indentation when it is formatted, the text block following it will begin at the indent position on the following line. For instance:

```
.hi "Special Uses:" -10
Form Letters, Table of Contents, Index, Footnotes
```

would format as

Special Uses:
Form Letters, Table of Contents, Index, Footnotes

Note that in this example, the first argument includes a space and therefore must be enclosed in quotes to enable TEXT to find the beginning of the second argument.

In our sample.fle, find the list of sections again and replace the break line (.br) instructions with hanging indent instructions. Let's have two spaces after each number instead of one. (You can accomplish this with the instructions ".hi 1. -4", ".hi 2. -4", etc.)

Remember to delete the characters in the hanging indent from the line following the instruction. For example, this is **WRONG:**

```
.hi 1. -4
1. Introduction: What Is TEXT and How Does It Work
```

and will format as

1. 1. Introduction: What Is TEXT and How Does It Work

Run TEXT again on the sample.fle when you have all the hanging indent instructions inserted. The list is now easier to understand at a glance as well as more attractive.

However, in many instances, you will want to line up the left edge of your hanging indents with your current left margin, rather than having them protrude into the margin space. This may be

accomplished by first indenting the same number of spaces as the second argument to your ".hi" instruction will specify for your hanging indents, and then returning the indentation to the previous level after your hanging indents. For instance, in our current example, we could line up the numbers of our list with the left margin of the paragraph that follows it by using

```
.in +4
.hi 1. -4
Introduction: What Is TEXT and How Does It Work
...
.hi 7. -4
Changing the Defaults: Patches
.in -4
```

Go back into the sample file and add the indenting instructions necessary to line the left edge of the hanging indents up with the left margin of the paragraph below them. Run TEXT again and notice how the format of the list has changed.

CHANGING THE INDENT OF A SINGLE LINE

If you have only one line that you want indented, TEXT offers you several ways of specifying it. You can just indent the line in your file the amount you want TEXT to indent it during formatting, since, as we have seen with our sample.fle, TEXT preserves all spaces to the left of any text on an input line. This, of course, will only work if the line is to be moved to the RIGHT of the left margin.

Two ".in" indenting instructions can be used as we have previously discussed, to move the left margin in one direction before the line and then the same amount in the opposite direction after it. This will work for changing the indentation in either direction, but TEXT will not fill or justify the line so it will not work properly for changing the indentation of the first line of a paragraph.

A better way to change the indentation of a single line is to use the temporary indent ".ti" instruction. This instruction changes the left margin for one line only after which TEXT automatically restores the previous indent level. This means you need only a single instruction to specify that a section heading, for instance, is to begin several spaces to the left of the left margin.

The temporary indent instruction works very much like the .in instruction. You must specify as an argument to the ".ti" instruction where you want the next line (only) to begin. You can specify an absolute character position, such as

```
ti 10
```

which will begin the line following it ten spaces from the left edge, regardless of where the left margin is set. You can also specify a relative position for the indentation, such as

```
.ti -8
```

which will start the line following it eight spaces to the LEFT of the current left margin setting, or

```
.ti +8
```

which will start the line following it eight spaces to the RIGHT of the current left margin setting. In all cases, the second line will begin at the current left margin setting.

Edit the sample.fle now and find the section entitled, "HOW DOES IT WORK?". The second line in this section may be considered a subheading, so let's move it to the left edge of the file and insert a temporary indent instruction that hangs it 6 spaces into the left margin:

```
.ti -6
1. TEXT formatting instructions
```

Since we know at a glance to the top of the file that our left margin setting at this point is 10, we can obtain the same result with the second subheading by using the absolute character position of 4 (i.e. 10 minus 6) in the .ti instruction:

```
.ti 4
2. The TEXT command
```

Do that now in the sample.fle. Run TEXT (type "text sample.file" at the operating system prompt) and note how the subheadings are formatted.

PARAGRAPHS

A special version of the ".ti" instruction is the paragraph instruction, ".pp", which will indent the line following it 5 spaces to the right. In addition to this ".ti +5", the ".pp" instruction also checks to see that there is room on the page for at least two lines of the paragraph. If there is not, the paragraph will begin on the next page since it is considered bad formatting practice to have the first line of a paragraph stranded alone on the bottom of a page.

example,

```
.pp
This is a paragraph that I want indented the
standard 5 spaces, with all lines including the
first filling normally.
```

format as

> This is a paragraph that I want indented the standard 5 spaces, with all lines including the first filling normally.

and the first two lines will never be split across a page

In the sample.fle, add a paragraph instruction to indent the paragraph that immediately follows the first subheading. It should look like this:

```
.pp
The first step is to put TEXT instructions
```

If you do not want your paragraphs indented, but you do want TEXT to keep the first two lines of a paragraph together, you can use the need ".ne" instruction to tell TEXT that it needs room for at least two lines on the page or must begin this paragraph on the next page.

```
.ne 2
This paragraph will not be indented, and
will not be broken between the first
and second lines.
```

The ".ne" instruction may be used to specify any number of lines that must be kept together on a page. For instance, if you have a five-line table that you do not want split across two pages, you can inform TEXT of this with:

```
.ne 5
table entry one
table entry two
...
```

When TEXT encounters the instruction, it will cneck to see if there is still room on the page for the specified number of lines. If there is, it will continue normally. If there is not, TEXT will leave the remaining lines blank and continue formatting the following text at the top of the next page.

Rather than inserting a ".ne" instruction before every paragraph in your document, you may want to run TEXT on your input file, review it on the screen, and then add the instruction only where it is needed. (See the subsection on Polishing Your Formatted Document for more information.)

When we last ran TEXT on our sample.fle, item 5. of our sections list was split at the bottom of page 3. Edit the file and insert the necessary TEXT instruction to keep both lines of item 5. together.

You should have inserted a line containing ".ne 2" into sample.fle just above the ".hi" instruction for item 5. This will force TEXT to put both lines of that item onto page 4.

You may have also noticed that page 4 ends with the first line of a paragraph. When TEXT acts on the ".ne" instruction we just inserted, the bottom line of every page that follows it will be forced onto the following page until a blank bottom line is found. However, just for practice let's add the correct ".ne" instruction above the paragraph that talks about Section 7 to insure that its first line is not stripped off again.

Run TEXT and note where each page now breaks.

PAGE HEADERS

Let's now return to the page header feature which was mentioned briefly near the beginning of this tutorial (see page 7, PAGE HEADER SPACE). You will remember that we discussed the header space instructions, ".h1" and ".h2", which control the amount of space at the top of each page. There are two instructions for this instead of just one because TEXT allows you to specify a one-line header at the top of your page, and the two instructions allow you to control its placement within the header space. The ".h1" instruction specifies the number of lines from the top of the page to and including the header. The ".h2" instruction specifies the number of lines from the header to the body of the page. TEXT initially sets the ".h1" to 3 and the ".h2" to 2 so that any specified header will appear as the center line of the 5 line header space.

The one-line header has three parts: a left adjusted part, a centered part and a right adjusted part. A possible header for our sample file might be:

Gillogly Software TEXT Introduction

The three parts, or strings, of this header can be specified with the ".he" header instruction in TEXT as follows:

"Gillogly Software"TEXT"Introduction"

The first string (i.e. Gillogly Software) will be placed flush left at the left margin setting that was in effect when the .he instruction was given. The third string (i.e. Introduction) will be placed flush right at the right margin setting in effect when the .he instruction was given. The second string will be centered on the line between those same left and right margin settings.

You may leave out any of the three header parts by leaving that field empty in the ".he" instruction. For instance, for a header that consists of only the right justified field, you would use the ".he" instruction as follows:

""Introduction"

For a centered section only, use

.he ""TEXT""

or, for left and right parts only, use

"Gillogly Software""Introduction"

TEXT will use the first character encountered after the ".he" as the string separator, so that if you want to use the double quote (") in the header you may replace it with any other character. For instance,

*Gillogly Software*TEXT*Introduction*

or

he jGillogly SoftwarejTEXTjIntroductionj

will work just

"Gillogly Software"TEXT"Introduction"

The specified strings may be any length from zero characters to the entire line length so long as the total length of the three strings together does not exceed that line length (the one in effect at the time the .he instruction is given).

The specified header will appear on every page that begins after the ".he" instruction. The header space and header are usually specified at the beginning of a file, but if you do not want a header on your first page, you may place the header instruction anywhere before the input for the second page begins but with at least one non-instruction line (even a blank one) above it in the file.

Let's go into our sample.fle now and add this header instruction at the beginning. Since we do not want it to show up on our first page, we'll insert a blank line and the header instruction as the last TEXT instructions before the text of the document begins.

```
(blank line)
.he "Gillogly Software"TEXT"Introduction"
TEXT USER'S MANUAL
```

Now we'll instruct TEXT to place our header two lines down from the top of each page and one line above the body of the text. We can do this by changing our header space instructions to

```
.h1 3
.h2 1
```

Run TEXT now and have the formatted document stored as file sample.txt (type text sample.fle sample.txt at your operating system prompt). Note the header that appears at the top of each page after the first.

PAGE FOOTERS

As with headers, you may also specify a one-line three-part footer to appear in the footer space at the bottom of each formatted page. The ".f1" and ".f2" instructions (discussed under PAGE FOOTER SPACE on page 8) control the amount of footer space and the position of the footer within that space. The ".f1" instruction specifies the number of lines from the bottom of the text to the footer. The ".f2" instruction specifies the number of lines including the footer to the bottom of the page. TEXT initially sets the ".f1" to 2 and the ".f2" to 3 so that any specified footer will appear as the center line of your 5 line footer space.

The footer is specified with the ".fo" instruction. Like the header, it may include a left-adjusted string, a centered string, and a right-justified string. As with the header instruction, TEXT will consider the first character after the ".fo" to be the separation character between strings. In most examples, we use the

double quote ("), but you may use any character you find convenient. A possible footer for our sample file might be

TEXT 4.0 March 1982 TEXT 4.0

We would specify this with the footer ".fo" instruction

.fo "TEXT 4.0"March 1982"TEXT 4.0"

Edit the sample.fle now and add this footer instruction to the beginning of the file near the footer space instructions. Remember to leave the blank line above the ".he" instruction. Don't put the footer instruction on it, even though it looks inviting. The footer will appear at the bottom of every page beginning with the one being formatted when the footer instruction is encountered.

While you're editing sample.fle, also change the footer space instructions so that there are two blank lines between the body of the text and the footer, and two lines including the footer to the bottom of the page. The new footer space instructions should be

```
.fl 2
.f2 2
```

Run TEXT again and store the formatted document in the same sample.txt file (text sample.fle sample.txt)

IN-LINE INSTRUCTIONS

It is now time to look at the second type of TEXT instruction, the "in-line" instruction. These instructions are two or three characters long and ALWAYS begin with a backslash (\). (If you have a keyboard modified for a foreign language, see Section 7 on Changing The Defaults: Patches). Whenever TEXT encounters a backslash in the text it is processing, it assumes that this character is introducing an in-line instruction. Since this backslash is then discarded by TEXT, if you want a backslash in your text output you must enter it as two backslashes (\\). TEXT will throw the first one away, will realize this is NOT an in-line instruction and will print the second backslash.

The in-line instructions may be placed anywhere on a text line or on a line by themselves. They may also be included as a part of a string or argument to one of the "separate line" instructions, such as the hanging indent ".hi", header ".he", or footer ".fo" instructions.

PAGE NUMBERS

You can have TEXT print the current page number by using the "in-line" page number instruction "\#". Whenever TEXT encounters the "\#" it will replace it with the number of the page being formatted. For instance, this sentence appears on page 18, which was obtained by entering

this sentence appears on page \#

Page numbers are especially useful in headers and footers. TEXT will number your pages automatically if you use a header instruction such as

```
.he ""-\#-""
```

which will give you a centered page number at the top of your page

-19-

or a footer instruction such as

```
fo "TEXT 4.0""\#"
```

which will supply the current page number at the bottom right corner of each page along with the string "TEXT 4.0" at the bottom left.

In our sample.fle, change the header ".he" instruction so that the center field is the current page number:

```
.he "Gillogly Software"\#"Introduction"
```

BEGIN A NEW PAGE

When you want TEXT to begin a new page, use the break page ".bp" instruction. For instance, if you want Chapter Two to begin a new page, you must insert

```
.bp
```

between the end of Chapter One and the beginning of Chapter Two TEXT will begin a new page as soon as it encounters the instruction

If you want the new page to begin at a different number, you may specify the number of the new page as an argument to the .bp command. For instance, suppose that after your Introduction you want Chapter One of your document to begin on page 5. You would add the instruction

```
5
```

to your input file just before Chapter One begins. If you have specified the page number as a part of your header or footer, the first page of Chapter One would be numbered page 5.

If you want TEXT to begin a new page and skip several page numbers, this can also be specified as an argument to the ".bp" instruction. For instance, you may want TEXT to skip four pages which you will use for figures and tables. You would specify this as

```
.bp +4
```

If TEXT is formatting page 5 when it encounters this instruction, the text which follows it will be formatted onto page 10 (skipping the four pages which would have been numbered 6, 7, 8 and 9).

In our sample.fle, have TEXT break the page so that the section, HOW DOES IT WORK?, begins on a new page. The input should look like this:

```
.bp
HOW DOES IT WORK?
```

Let's also break the page after the subsection on the TEXT command, and skip 10 page numbers just for fun. The last paragraph in that subsection is

```
     Additional functions available
from the command line are described in
Section 4.
.bp +10
```

Run TEXT on the sample.fle and check the page numbers in the header on each page. Did HOW DOES IT WORK? begin on a new page? Did the numbering skip 10 pages after the TEXT command section (the page numbers should be 11 apart)?

LINE SPACING

For those documents you want to print double or triple spaced, TEXT provides the line spacing ".ls" instruction. Normally TEXT will format your document single spaced, but if you want it double spaced instead, simply add

```
.ls 2
```

to the begining of your input file. Similarly

```
.ls 3
```

will triple space it. If you want only a small section of the document double spaced, insert the double spacing instruction

```
.ls 2
```

immediately above the text to be double spaced and return to single spacing with

```
.ls 1
```

immediately following it.

Let's add the instructions to our sample.fle to output it double spaced, but with the list of sections single spaced. (Add ".ls 2" to the list of initial TEXT instructions at the top of the file, ".ls 1" above the first hanging indent instruction of the list of sections, and ".ls 2" again after the last item of the list.)

Run TEXT and review the output on your screen

ADDING COMMENTS TO YOUR INPUT FILE

There is a special in-line instruction that tells TEXT to ignore everything that appears to the right of it on the line. This comment instruction `\"' is used for marking everything in your document file that you want TEXT to ignore during processing. For instance, you might want to add a comment to an entry in a table you are printing to remind yourself where that entry came from in case you need to change it.

```
.nf
entry 1 of table B      \"Total from line 39
entry 2 of table B      \"Total from line 45
entry 3 of table B      \"Same as entry 5 of table A
```

If you need to change entry three of this table, you have a reminder right there that you must also change entry five on table two. When this table is formatted by TEXT, the three entries will appear, but the text to the right of each comment instruction will not:

```
entry 1 of table B
entry 2 of table B
entry 3 of table B
```

You can also use it to add a comment to a TEXT instruction:

```
rm 110  \"this setting for 130-column paper
```

If you want an entire line ignored, you must precede the `\"' with a period (.), placed in column one as with other separate line instructions. For instance, you may use these two versions of the comment instruction to delete something from your output without removing it from the document:

```
This sentence will remain in my document. \"But
.\"this sentence will not appear in the formatted
.\"version, although it will remain here.
This sentence will also appear.
```

When TEXT encounters this paragraph it will format as

> This sentence will remain in my document. This sentence will also appear.

The comment instruction is also useful for TEXT instructions you must change back and forth:

```
.rm 120     \"this setting for 130-column paper
.\".rm 70     \"this setting for 80-column paper
```

In this example, when you change your printer paper from 130-column to 80-column, you need only remove the `\"' from the second instruction and add it to the beginning of the line on the first instruction:

```
.\".rm 120     \"this setting for 130-column paper
.rm 70      \"this setting for 80-column paper
```

When TEXT encounters the first set of instructions it will set the right margin to 120 and completely ignore the second line. When it encounters the second set of instructions, it will completely ignore the first line and set the right margin to 70 as instructed on the second line.

Edit our sample.fle now, and use comment instructions to disable the double spacing instructions we inserted at the beginning of the file and around the list of sections.

```
.\".ls 2
```

Let's also add a comment to the line that demonstrates the command to use for printing formatted documents on a printer:

```
>text sample.fle lp:    \" send output to printer
```

Run TEXT to see that the double spacing really was disabled, and that all the comments were ignored.

EMBOLDENING

In many of your documents, you will want to give special emphasis to certain words or phrases, and have such things as titles, headings and subheadings print darker than the rest of the text. TEXT provides the bold ".bd" instruction as one way of accomplishing this. With the

```
.bd
```

instruction TEXT will print the line which follows it in the input file darker than the preceding or following text. TEXT produces this emboldening by overstriking the indicated characters a second time. When the formatted text is displayed on a Z89/H89/H19 screen, the emboldened characters appear in inverse video.

The characters on the input line may be a single word or an entire sentence, and may occur in the middle of a sentence such as this one which was obtained with the following input:

```
... such as this
.bd
one
which was obtained with the following input:
```

Normally TEXT overstrikes each emboldened character once, but you may specify a different number of overstrikes with the ".bn" instruction. This number is specified as an argument to the ".bn" instruction and must appear in the input file before the first ".bd" instruction you want affected by it. For instance, if your printer ribbon is getting a little tired and you want TEXT to overstrike all your bold requests three times instead of the usual one, use

```
3
```

at the beginning of your input file. To increase or decrease the

number of overstrikes for emboldening, you can also give a relative number as the argument to the ".bn" instruction:

```
.bn +2
```

will increase the number of overstrikes by two, no matter what it was set at before. Likewise

```
.bn -1
```

will decrease the number of overstrikes for emboldening by one.

To return to the default single overstrike, just insert the ".bn" instruction without any argument:

```
.bn
```

If you want to embolden more than one line you may specify the number of input lines to be emboldened as an argument to the ".bd" instruction. For instance, to have the three lines following the ".bd" instruction emboldened, use

```
.bd 3
```

The text on the fourth line following the instruction will again print normally.

```
words words words in current paragraph
.bd 3
word or words to be emboldened;
more words to be emboldened;
and still more.
Continuing text which will NOT be emboldened.
```

If you want a whole paragraph to be emboldened and don't want to hassle with counting the number of input lines this involves, you can specify

```
 bd 999
```

at the beginning of the paragraph, and

```
.bd 0
```

at the end

In our sample.fle, let's now add instructions to have the title at the top of the page printed in bold, overstriking four times to make it very dark. Let's then return to the default single overstrike for the rest of the document.

```
.bn 4
.bd
TEXT USER'S MANUAL
.bn
```

Let's also have the entire last paragraph of the subsection on the TEXT command emboldened.

```
.bd 999
     Additional functions available
from the command line are described in
Section 4.
.bd 0
```

Using ".bn 1" or ".bn -3" in place of the ".bn", and using ".bd 3" in place of the ".bd 999" and ".bd 0" will obtain the same results.

Run TEXT and note the way the emboldened characters are displayed on your screen. If you do not have a Z89/H89/H19, or if you do not want your emboldened text displayed in inverse video, you may disable this feature by typing "-t" at the operating system prompt between "text" and the name of the file:

```
>text -t sample.fle
```

Special features that you can specify when running the program are called flags. The "-t" is a flag to the TEXT command. Run TEXT again, this time with the "-t" flag. The inverse video feature may be disabled permanently by patching the command file (see Section 7, Changing The Defaults: Patches).

The separate line ".bd" instruction works very well for single lines and filled or justified text. However, if you want to embolden a single word in a line that is being formatted in no-fill mode, you need a way to specify emboldening without disrupting your input lines.

TEXT provides this capability with its in-line emboldening instructions, "\<" and "\>". The "\<" instruction tells TEXT to begin emboldening and the "\>" instruction tells TEXT to discontinue it. Once TEXT encounters the "\<" it will begin overstriking everything until it is told to stop with a "\>" or a ".bd 0" instruction. As with the ".bd" instruction, it will overstrike each character once unless the ".bn" instruction has been used to change this value.

```
\<     begin emboldening
\>     end emboldening
```

For instance, to embolden the word "Patches" in the following line, you could enter the instructions as

```
Changing the Defaults: \<Patches\>
```

to get

```
Changing the Defaults: Patches
```

In our sample.fle, using the in-line emboldening instructions, have TEXT overstrike the word "TEXT" in the two numbered subheadings:

```
1. \<TEXT\> formatting instructions
```

and

```
2. The \<TEXT\> command
```

The in-line emboldening instructions are also essential to get emboldening in a hanging indent, header or footer. Let's embolden the right-adjusted field of the header in our sample.fle:

```
he "Gillogly Software"\#"\<Introduction\>"
```

To see how our emboldening instructions are interpreted by TEXT, we will need to send the output of our sample.fle to the printer. Use the same command as you would for sending the TEXT output to a file for review but substitute the name of your printer for the review file name:

```
text sample.fle lp:
```

for HDOS, or, under CP/M:

```
text sample.fle lpt:
```

Run TEXT now and send the output to your printer. Note how the words we specified as bold are darker than the rest of the text. If there is not enough of a variation, you may want to go back into the sample.fle and increase the number in the ".bn" instruction to get additional overstrikes.

You may want to take this opportunity to review in hardcopy the results of our other formatting instructions.

PROBLEMS?

If you entered all the instructions correctly, and it worked on your screen but failed when you sent it to the printer, either in giving you improper page breaks or in emboldening improperly, perhaps printing the overstrikes on the line below instead, this section is for you.

Page Break Problems

If you are getting blank pages interspersed during printing or pages breaking after too few lines, your printer driver is probably trying to control the page length and conflicting with the page length set within TEXT. This may be remedied by typing the following at your operating system prompt:

```
>set lp: page 0
```

[This problem and remedy only apply to systems running under HDOS.

Emboldening Problems

For overstriking on a printer, TEXT prints the entire line, returns to the left margin without a linefeed (so that the paper does not advance), then sends the overstriking characters for that same line. Some printers do not have the capability to do more than one pass across a line, but can backspace a character at a time. For these printers there is an instruction which tells TEXT to send each overstrike to the printer as a character followed by a backspace

followed by the overstrike character followed by the next character on the line. This is much slower than the default used by TEXT, but if your printer is one that requires overstriking to be done in this backspace mode, you must let TEXT know with the ".bs" instruction. The instruction

```
.bs 1
```

turns this mode on. You will want to include it at the top of every file. If you ever need to,

```
.bs 0
```

will turn the backspace mode off again and return you to the default method.

For those printers that do not have backspacing capability either (such as the Comprint 912), you will be unable to use the emboldening, underlining or overstriking capabilities of TEXT with your machine.

UNDERLINING

Another way of providing special emphasis in your documents is the use of underlines. TEXT provides two different types of underlining: word underline, which underlines each word and adjacent punctuation, but not associated spaces, and continuous underline, which gives you a solid underline from beginning to end including all spaces.

This is an example of word underlining.

This is an example of continuous underlining.

Let's look first at the word underline instructions. These work just like the emboldening instructions. The separate line instruction is

```
.ul
```

All characters on the line immediately following this instruction will be underlined, including punctuation and special characters such as *, &, #, and %. This line, for instance, was formatted with the instruction:

```
.ul
This line,
for instance, was formatted.
```

If you want to underline several lines, you may give the number of input lines to be underlined as an argument to the ".ul" instruction, as with

```
.ul 3
```

to underline the words on the next three lines, or, if you do not want to bother counting lines, you may specify a suitably large number such as

```
ul 999
```

to begin word underlining and

```
ul 0
```

to end it.

In our sample.fle, let's specify word underlining for the first example using the TEXT command in section 2 of "HOW DOES IT WORK?".

```
ul
      >text sample.fle
```

Let's also underline every word in the last sentence of the very first paragraph:

```
.ul 2
It will even create a table of contents and index,
and it's invaluable for form letters.
```

You may use the ".ul 999" instead of the ".ul 2" instruction, if you follow the second line with the ".ul 0" to end the underlining.

To get continuous underlining, where spaces are also underlined, use the instruction

```
.cu
```

If you have several lines that you want continuously underlined you may count them and specify the number as an argument:

```
.cu 3
```

or you may use the

```
.cu 999
```

to begin continuous underlining and the

```
.cu 0
```

to end it.

In our sample.fle, let's specify continuous underlining for the first sentence of the section titled, HOW DOES IT WORK?.

```
.cu
Using TEXT to format a document requires two steps
```

Let's also underline the entire paragraph that follows the subtitle.

```
.cu 999
The first step is to put TEXT instructions.

such as PIE, EDIT, or
.cu 0
```

If you run TEXT on the sample.fle now with the command

```
text sample.fle
```

the words to be underlined will be displayed in inverse video on the screen on H89, Z89 and H19 terminals. Notice the difference between the word underline and the continuous underline? (For non-H89/Z89/H19 terminals or to disable this inverse video display feature, use the -t flag as described under EMBOLDENING, page 24.)

As with emboldening, TEXT also provides "in-line" instructions for word and continuous underlining. You may find them easier to use and they are essential for underlining a word without underlining the adjacent punctuation and for underlining parts of unfilled text. For word underline, use the instructions "\{" to turn it on and "\}" to turn it off. For continuous underlining, use "\[" to turn it on and "\]" to turn it off.

\{	begin word underlining
\}	end word underlining
\[	begin continuous underlining
\]	end continuous underlining

Using the in-line instructions, specify word underline for the titles of the books in the references at the end of the sample.fle, but do not underline any punctuation:

```
TEXT is based on and greatly extended from \{format\} by
...
Kernighan, B.W. and Plauger, P.J.; \{Software Tools\}
```

Let's specify continuous underlining for the phrase:

```
centering instruction, ".ce"
```

which occurs in the last line of the subsection titled "1. TEXT formatting instructions"

```
the \[centering instruction, ".ce"\] preceding the title
```

Run TEXT and direct the output to your printer. Check your results. Did the underlining work as you expected? Note the differences between the two types of underlining.

CHANGING THE UNDERLINE CHARACTER

As you have noticed on your printout, TEXT uses the underscore "_" to underline specified characters. For those times when you want to use another character, such as the hyphen "-", TEXT provides the underscore character ".uc" instruction. The character you want substituted for the underscore is given as an argument to the ".uc" instruction. For example, to substitute the hyphen, use

```
.uc -
.ul
This text will be crossed out.
```

which will format as

~~This text will be crossed out.~~

Or, if you do NOT want the punctuation included:

```
.uc -
\{This text will be crossed out\}
```

will produce

~~This text will be crossed out~~.

Similarly, the continuous underline instructions ".cu", "\[" "\]" will produce

~~This-text-is-crossed-out-using-continuous underline-instructions~~.

To return to the underscore character for normal underlining, simply give the instruction without an argument:

```
.uc
```

In our sample.fle, add the instructions to cross out parenthetical phrase in the second paragraph:

```
.uc -
.ul 2
(11" paper is 66 lines, but you could
specify 84 lines for legal size 14" paper instead)\}
.uc
```

NOTE in the example above that the in-line instruction "\}" was used to turn off the underlining (crossing out in this instance) before the punctuation was encountered. If you neglect to include the ".uc" instruction to return the underlining character to the underscore, all underlined text in the rest of the document will be crossed out instead.

OVERSTRIKING

For those times when you want to overstrike one character with another, TEXT provides the backspace "\b" instruction. This is inserted in the middle of the word immediately following the character to be overstruck and immediately followed by the overstrike. For instance, the French words

l'école française

are produced by overstiking the "e" with the accent "'" to produce "é" and by overstriking the "c" with the comma "," to produce "ç", thus:

l'e\b'cole franc\b,aise

Each "\b" will be interpreted as a single backspace making multiple overstrikes with different characters possible. Normally TEXT will send the line to the printer without backspaces or overstrikes, will then return to the left margin without a linefeed (to avoid advancing the paper), and will then print all the overstrikes in the next pass across the line. If your printer does not have the capability to do a second pass over a line but can do character backspaces, you must use the ".bs 1" instruction as described in the PROBLEMS? section on page 25. TEXT will then embed the backspaces in the output line in the correct positions to obtain the specified overstrikes. This is slower than the default but the end results are the same.

In our sample.fle, let's add to the second sentence of the first paragraph the words,

to produce foreign words such as the Welsh
môr-leider (pirate) or Danish høne (chicken).

by typing

to produce foreign words such as the Welsh
mo\b^r-leider (pirate) or Danish ho\b/ne (chicken)

If you run TEXT and review the output on your screen, you will notice that only the overstrike character appears. For instance, høne displays as h/ne. Since your screen cannot display more than one character in a space at a time, it displays the last character sent to that space. When you run TEXT and send the output to your printer (text sample.fle lp: or text sample.fle lpt:) you will see all the specified characters printing in their corresponding spaces. Run TEXT now to see how the overstrike capability works.

CENTERING

In many of your documents you will want to center a line over the body of your text. The ".ce" centering instruction will center the line following it on your current line length between the current left indent and right margin settings.

```
.ce
centered line
```

will produce

centered line

If you want to center more than one line, you may specify the number of lines to be centered in the ".ce" instruction. For instance,

```
3
```

will center the next three lines. Since blank lines are included in this count, you may want to use the same technique for a large number of centered lines as you did with underlining and emboldening. Specify a large number with the ".ce" instruction, such as

```
999
```

and then follow the text to be centered with a

```
.ce 0
```

to discontinue centering. This will be useful for title pages, for example, where you may have several centered lines with lots of blank space around them.

In our sample.fle add an instruction to center the title. Run TEXT and review the output on the screen. The emboldened words "TEXT USER'S MANUAL" should now be centered between the indent setting (10) and the right margin setting (79).

ADDING BLANK SPACE

For each blank line you leave in your input file, TEXT will give you a blank line in your output. For larger numbers of blank lines, it is easier to use the space ".sp" instruction. For instance, if you want 25 blank lines between the title and author lines on your title page, you can just insert

```
.sp 25
```

between the title and author lines:

```
title
.sp 25
author
```

If you give the ".sp" instruction without an argument

```
.sp
```

TEXT will give you a single blank line

When you specify more blank lines than there are remaining on the page TEXT is formatting, either with the ".sp" instruction or by leaving blank lines in your input file, TEXT will give you blank lines to the end of the page but will not carry them over to the top of the next page. For instance, if TEXT encounters the instruction ".sp 25" when there are only 20 lines left on the page, those 20 lines will be left blank and the other 5 requested will be disregarded. TEXT will then continue formatting at the top of the next page.

Let's now add a title page to our sample.fle with the title about 12 lines down from the top of the page. Since we have set our header space (".h1" and ".h2") to 4 lines and we have inserted a blank line above the ".he" header instruction to prevent it from printing on our first page, we will need to add a space of 7 additional blank lines. Add the space ".sp" instruction to the end of our initial instructions, but before those instructions that deal specifically with the first line:

```
.sp 7
.bd
.ce
TEXT USER'S MANUAL
...
```

We'll title the document A GUIDE TO FORMATTING WITH TEXT and include a date centered six lines below this. Let's also add an instruction to have the text of the document begin on the next page, and have that page numbered page one.

```
.sp 7
.ce 100
A GUIDE TO FORMATTING WITH TEXT
.sp 5
March 1982
.ce 0
.bp 1
.bd
.ce
TEXT USER'S MANUAL
...
```

(Did you notice that we could remove the centering instruction that appears directly above "TEXT USER'S MANUAL" on the first line of the second page and move the ".ce 0" instruction directly below it to get the same result with less instructions?)

Run TEXT and review the title page on the screen. (The lines on the title page will appear slightly off-center, because they are being centered between our current margin settings of 10 and 79.) Did the header for the second page give the page number as 1?

STRING SUBSTITUTIONS

TEXT provides a shorthand instruction for words and phrases that may recur many times in a document. With this instruction, you may assign a single-letter name to a set of characters which may include in-line TEXT instructions. The ".sb" instruction has two parts (arguments): the single letter name and the set of characters (the string) to be substituted for it.

```
.sb x set of characters to be substituted
```

Once you have defined a string substitution with the ".sb" instruction, each time you want that set of characters in your document, you need only use the in-line substitute instruction "\s" and the single-letter name that corresponds to that string (in this case \sx) and TEXT will substitute the set of characters for it during formatting.

For instance, if you are doing a review of the book, La Mère de la Marquise you could use the ".sb" substitute instruction to assign the letter "t" to the underlined title

```
.sb t \{La Me\b`re de la Marquise\}
```

and the letter "w" to the author's name

```
.sb w Edmond About
```

Then within your text, every time you want the underlined title to appear you simply input \st, and every time you want the author to appear you use \sw. Text will do the substitutions automatically.

```
This is a review of \st by \sw. \sw was born at
Dieuze, in Lorraine, on Februrary 14, 1928. \st was
one of his later works and clearly demonstrates his
literary style.
```

will produce

> This is a review of La Mère de la Marquise by Edmond About. Edmond About was born at Dieuze, in Lorraine, on Februrary 14, 1928. La Mère de la Marquise was one of his later works and clearly demonstrates his literary style.

TEXT allows you to define up to 26 different string substitutions using the lower case alphabet.

In our sample.fle let's assign the letter "g" to the string "Gillogly Software" and let's make it bold:

```
.sb g \<Gillogly Software\>
```

So far this string appears only in our header, so we can change our header instruction to

```
.he "\sg"\#"\<Introduction\>"
```

This will make changing the right field of the header easier to specify when we begin Chapter One since we will not need to check back to see exactly what we put in the left field but only enter "\sg" and let TEXT remember spellings, emboldening and such.

```
.he "\sg"\#"\<Chapter One\>"
```

Run TEXT and send the output to the screen for review. Note that the header still has Gillogly Software at the left corner and that it is now emboldened.

If you need more than 26 substitutions, TEXT allows you to reassign a particular letter to a new string at any time. This will also be very useful in form letters and other repetitive documents where only a few words change from one printing to the next. For instance, you can set up the inside address on a form letter with string names. Then, when you run 20 copies of the letter, you can just have TEXT reassign those strings to a new name and address between each copy (see Section 5 for details).

CAUTION: The memory TEXT uses for remembering string substitutions is not recycled, so very large numbers of substitutions will eventually result in an "Insufficient memory" error.

Additional uses for this instruction are described in Chapter 5, Special Uses: Form Letters, Table of Contents & Index, Footnotes.

INCORPORATING OTHER FILES

One of the instructions that adds tremendous flexibility to TEXT is the read file ".rf" instruction which requires TEXT to insert the contents of another file into your document during formatting. As with the substitute string instructions ".sb" and "\s" where TEXT substitutes a string of characters for the string name at the point in your document where the string name is encountered, TEXT will substitute an entire file for the line containing the ".rf" instruction. All TEXT instructions and text in the second file will be processed and formatted as if the entire second file were part of the first file.

The name of the file to be included is specified with the ".rf" instruction

```
.rf filename
```

For instance, you may have figured out all the TEXT formatting instructions and text necessary to output a lovely centered, emboldened and well-spaced name and address that you want to use for all your letters. Every time you write a letter your first 14 lines are

```
.sp 2
.ce 100
.bn 6
.bd 100
GILLOGLY SOFTWARE
```

```
.bn 3
2520 Chard Avenue
Topanga, CA 90290

.bd 0
(213) 455-1407
.ce 0
.sp 3
```

Rather than having to enter all this information every time, you could put these 14 lines into a letterhead file called letter.hd and then begin every letter with only the one line

```
.rf letter.hd
```

TEXT will process the instructions and text in letter.hd just as if they were all right there in your letter file. In either case, your final letter will look the same.

Suppose you also have a set of formatting instructions that you use at the beginning of most of your documents to set your margins, page length, headers and footers. To save yourself time and effort, and to insure that all the documents begin with exactly the same formatting instructions, you could put all the instructions in an initialization file called setup and begin every document file with

```
.rf setup
```

If your letters use the same initialization instructions you can begin your letter files with

```
.rf setup
.rf letter.hd
```

There is no limit to the number of ".rf" instructions you may have in a file, and a ".rf" file may itself include ".rf" instructions to files which include ".rf" instructions to files, etc. so long as you have no more than four ".rf" files active (open) at one time.

TEXT will look for the named file on the drive where the file containing the ".rf" instruction is located. If you are using HDOS and TEXT cannot locate the file there, it will check for it on all other mounted drives. If it still cannot find the file it will print this message on the screen:

```
Could not find file filename on any drive.
Do you want to
        0: reset SY0: with a new disk
        1: reset SY1: with a new disk
        2: reset SY2: with a new disk
        A: reset DK0: with a new disk
        B: reset DK1: with a new disk
        C: reset DK2: with a new disk
        i: ignore this .rf command and continue
        n: specify a new file name
        a: abort the whole run
Action:
```

If you are using CP/M and TEXT cannot locate the named file on the drive where the file containing the ".rf" instruction is located, it will give you the message:

```
Could not find file filename.
Do you want to
        try a different drive? If so, type the letter of the drive
        IN LOWER CASE. The diskette may be replaced first.
        I: ignore this .rf command and continue
        N: specify a new file name
        A: abort the whole run
Action:
```

You may then select from the listed options by typing the letter or number that corresponds to your choice. You may either provide a new disk upon which TEXT may search for the named file; provide a new file name for which TEXT may search; skip the current ".rf" file instruction; or abort the program. If you choose to provide a new disk and you specify a drive containing active (open) files, TEXT will ask if it is

OK to close files there (y or n)?

If you answer "n", TEXT will repeat the list of choices. You must type "y" to proceed.

This gives you the capability to have TEXT continuously format a multi-disk document by creating a master file of ".rf" instructions which names all the files to be included.

master.fle:

```
.rf chapter.1
.rf chapter.2
.rf chapter.3
...
.rf chapter.18
.rf chapter.19
.rf chapter.20
```

When TEXT encounters a file name it cannot find, it will allow you to mount the disk containing that file. You will want to leave the disk containing this master file mounted on one of your drives so that TEXT can return to it after processing each file indicated by a ".rf" instruction. Your supplementary disks may then be mounted on another drive.

If you are running on a single drive system, you will need to have a set of master files, one on each disk, which calls all the relevant files on that disk and finishes by calling the master file of the next disk.

master.fl1 on disk 1

```
.rf chapter.1
.rf chapter.2
.rf chapter.3
.rf master.fl2
```

master.fl2 on disk 2

```
.rf chapter.4
.rf chapter.5
.rf chapter.6
.rf master.fl3
```

You will then run TEXT with the command

```
text master.fl1
```

TEXT will process all the ".rf" files requested in master.fl1 until it reaches the last one (master.fl2) which it will not be able to find. It will then give you the choice of resetting your single drive to continue looking for that file.

When you input the new disk, it will find master.fl2 and continue reading and processing the files specified in the ".rf" instructions given there.

In our sample.fle, let's insert a ".rf" instruction to have the contents of the file sample2.fle inserted in our output on the title page.

```
...
March 1982
.ce 0
.rf sample2.fle
.bp 1
...
```

Run TEXT on the sample.fle to find out what sample2.fle contains

(If you examine the contents of sample2.fle, you will notice some TEXT instructions that are as yet unfamiliar to you. They will be explained shortly.)

WRITING TEMPORARY FILES DURING FORMATTING

In addition to incorporating other files during formatting, TEXT can also create files to be incorporated later in the document. Let us say, for instance, that you want to reference a previous page by number in your document:

> See page 34 for information on incorporating other files.

You could run TEXT on your document, check what page number the referenced information appears on and insert that page number here.

However, if you do any editing on the information between that page and the reference to it, the page number may change. Rather than having to remember to check and possibly change all your page number references every time you insert or delete a few lines, wouldn't it be better to have TEXT keep track of it for you automatically? You can have TEXT, as it is formatting the document, write a file containing the page number you want to reference as it is processing that page, and then have it read in the contents of that file further along in your document for your reference.

To have TEXT write a file, you use the write file ".wf" instruction with the name of the file you want to write to.

```
.wf refpg
```

Every line between this ".wf" instruction and an end of write file ".we" instruction will be diverted to the named file. If the file already exists, and this is the first ".wf" instruction to that particular file, TEXT will erase the file before writing in it. If this is not the first ".wf" to that file during the current processing, TEXT will add the diverted lines to the end of the file. If the file does not exist, TEXT will create it.

TEXT will not process any separate line instructions included between the write file instructions, but will simply copy them to the named file. They will be processed when they are read back into the document with a ".rf" instruction.

Any in-line page number "\#" or substitute string "\s" instructions will be interpreted as they are encountered and the specific page number or string will be inserted in the write file. All other in-line instructions to be included in a write file must be **double backslashed**, because TEXT will strip off the first backslash in its search for the "\#" and "\s" instructions, and will then copy what is left to the write file. Each double backslash will be copied to the write file as a single one and will be ready for processing when read back into the document with a ".rf" instruction. If you want a page number or string substitution to be interpreted as it is read back in, instead of at the time it is diverted to a write file, specify it with a double-backslash also.

In our example above, you would insert the write file instructions

```
.wf refpg
\#
.we
```

in your document when the material you want to refer back to is discussed. TEXT will create a file called "refpg" that will contain a single line with the number of the current page TEXT is formatting. Then, further along in your document, when you are ready to reference the page, you would have TEXT read in that file:

```
See page
.rf refpg
for information on incorporating
other files.
```

Now TEXT will handle the reference automatically and you do not need to consider it again. The reference will always give the correct page number no matter how many times you move things around between the two.

If you want the page number to be printed in bold you could specify it with

```
.wf refpg
.bd
\#
.we
```

or by using double-backslashes with

```
.wf refpg
\\<\#\\>
.we
```

The write file instructions were designed with creating a Table of Contents in mind. You can use the write file instructions to create a file for your Table of Contents, such as tbl.con, and then have TEXT add the current section name and page number to this file each time you begin a new section. For example,

```
.wf tbl.con
.bp 3          \\"number the page 3
.ce
TABLE OF CONTENTS
.sp 3
.nf
Section One Title                    \#
.we

...three pages of text.

.wf tbl.con
Section Two Title                    \#
.we
```

The file tbl.con will look like this:

```
.bp 3          \"number the page 3
.ce
TABLE OF CONTENTS
.sp 3
.nf
Section One Title                    5
Section Two Title                    8
```

At the end of your document, after all the entries have been made to your table of contents, you can have TEXT process the file with the instruction

```
.rf tbl.con
```

To help with the formatting of your Table of Contents, TEXT provides the table of contents entry ".tc" instruction. This instruction,

like the header and footer instructions, has three fields which may be separated by any non-space character such as the double quote ("). The contents of the first field will be left justified against the left margin setting in effect during output; the third field will be right justified against the right margin setting; the single character specified in the middle field will be repeated to span the distance between the other two (unlike the ".he" and ".fo" instructions where the middle field is centered).

```
.tc "Section One Title "." 5"
```

for instance, will format as

```
Section One Title.....................................5
```

Any in-line instructions may be included, such as

```
.tc "\<Section One Title\>
```

to embolden the contents of the left field and insert the current page number in the right one. (The in-line emboldening instructions must be double-backslashed if this line is being diverted to a write file with the ".wf" and ".we" instructions.)

Although the instruction was designed for a table of contents entry, it has many other applications. Any of the three fields may be left empty, so you can use this instruction to draw a line from one margin to the other, for instance, with

```
.tc "" ""
```

or to right justify an inside address on a letter with

```
.tc """2520 S. Chard Avenue"
.tc """Topanga, CA  90290"
```

Create a Table of Contents for our sample.fle. Include an underlined, centered title and a centered page number footer.

```
.wf tblofc
.fo ""\\#""
.ce
\\[TABLE OF CONTENTS\\]
.sp 3
.we
```

Directly below the heading for each section, make an entry to the Table of Contents file so that the proper page numbers are automatically included. Use the ".tc" instruction and leave a blank space before and after the line of periods. For example, the entry for the last section in sample.fle, may be specified with :

```
HOW TO USE THIS MANUAL
.wf tblofc
.tc "How To Use This Manual "."
.we

This manual is divided into eight sections
```

Add a read file ".rf" instruction at the end of sample.fle to have the Table of Contents processed. We can precede it with an instruction to have the Table of Contents on a page by itself and numbered page 2:

```
.bp 2
.rf tblofc
```

Run TEXT on your sample.fle and have the output sent to your printer. Compare the page numbers listed in the Table of Contents to the beginning page numbers of each section.

NOTE: See Section 5, Special Uses: Form Letters, Table of Contents & Index, Footnotes for additional information and examples.

POLISHING YOUR FORMATTED DOCUMENT

When you have a document file ready for formatting, it is a good idea to review the final format before you actually print it. You can do this by reviewing the document as it scrolls by on your screen, using the "CTRL-S" and "CTRL-Q" capabilities of your terminal to stop and start the flow of text, or you can have TEXT store the formatted document in a review file with the command

```
text sample.fle review.fle
```

The review file will contain your formatted document with all the necessary printer control characters included so that you may simply print the file on your printer if you are satisfied with it as it is. These printer control characters and TEXT's overstrike capability may make some lines look a little strange, but they should not interfere with your review. Once you are satisfied with the content and general layout of your document, you can turn your attention to giving it a final polish, making sure lines and pages break appropriately.

MANIPULATING PAGE BREAKS

In professional text formatting, it is considered bad form to have the the last line of a paragraph as the first line at the top of a page (known as a widow) or the first line of a paragraph as the last line on a page (known as an orphan). You also do not want a heading or subheading appearing as the last line on a page. TEXT provides you with several ways of controlling where the pages are going to break.

We have already discussed the ".bp" instruction (see page 19) which may be used to force a page break at a specific spot in the text. However, this instruction will cause a line break (see page 11) in filling and justification, so it cannot be used to control the page break inside paragraphs.

We have also discussed using the ".pp" instruction (see page 14) to prevent the orphans that occur when a page breaks between the first

and second lines of a paragraph, and the ".ne" instruction which may be used to keep lines together on a page such as to prevent headings or subheadings from being separated from the text they head.

But what can you do if your review file shows a page that begins with the last line of a paragraph? If the paragraph is 8 lines long and you insert a

```
.ne 8
```

on the line above it, TEXT will leave seven blank lines at the bottom of the current page and begin the paragraph at the top of the next page. This is a bit drastic, and will look a bit odd. It would be better if you could either squeeze that last line onto the previous page, or force the next-to-last line onto the top of the new page to keep the last line company.

You can accomplish either of these remedies by changing your footer space designation for that one page. You can make room for that last line by shortening your footer space on that page only by one line. The instruction

```
.fl -1
```

will give you one less line between your text and your footer (if any). The instruction

```
-1
```

will give you one less line between your footer and the bottom of the page. It is best to place the instruction directly above the paragraph in question, and to place directly below the paragraph a companion instruction to return the footer space to its previous setting, either

```
.fl +1
```

or

```
.f2 +1
```

To force the next-to-last line of the paragraph onto the top of the following page, you need to expand your footer space so that there is no longer room for it. The instruction

```
.fl +1
```

will add one blank line between the body of your text and your footer. The instruction

```
+1
```

will add a blank line between your footer and the bottom of the page. To return the footer space settings to normal, insert the companion instruction after the paragraph: either

```
-1
```

or

```
.f2 -1
```

If you want the page break at the bottom of the second page to be unaffected by the addition or subtraction of a line at the top, you may make a temporary change to either the header space at the top or the footer space at the bottom of this second page. If you have expanded your footer space on the previous page to force an extra line onto the current page, you will need to shorten the header space or footer space on this page to make room for it. The instruction

```
-1
```

will eliminate one of the lines between the top of the page and your header (if any). The

```
h2 -1
```

instruction will eliminate a line between the header and the text. This instruction should also be inserted above the paragraph in question, with its companion instruction which will return the header space to normal, inserted below it, so that it will not take effect until the following page.

Run TEXT on the sample.fle now and have the formatted document stored in the file, review.fle:

```
sample.fle review.fle
```

Review the review.fle and pay particular attention to the page breaks. We have a bad break on page 16. Insert the necessary TEXT instructions to pull the line at the top of page 16 to the bottom of 15.

Run TEXT again and check the new page breaks.

MANIPULATING LINE BREAKS

If TEXT is breaking a line between two words you do not want split, you can use the unpaddable space "\ " instruction to force TEXT to keep them together on a line. TEXT will <u>never</u> split a line at an unpaddable space. For instance, if you never want TEXT to split the phrase "TEXT 4.0", you can indicate this by typing it in your input file as

```
TEXT\ 4.0
```

TEXT will treat the unpaddable space as if it were a letter in the middle of a word rather than a space between two words.

If you find during justification that TEXT is adding another space to the space between two words you do not want expanded, you may indicated this also with the unpaddable space. TEXT will <u>never</u> add another space to an unpaddable space when justifying.

HYPHENATION

In the review file you may also notice some justified lines that are very "thin" or filled, unjustified lines that are much shorter than those around them. Both these situations occur when the next word on a line that is almost full turns out to be too long for the space remaining. If TEXT were to hyphenate this word, the line would look much better.

You may indicate to TEXT where the acceptable hyphenation points in a word are by adding the characters "\-" at each of these points within the word. During processing, if such a word is too long for tne space remaining on the line, TEXT will hyphenate it at the optimal point. TEXT will also split hyphenated words, such as stand-alone, at the hyphen.

In our review.fle you will notice that the first line of the second paragraph contains a lot of spaces when it is justified. It would look more attractive if the word "articles" on the next line were to be hyphenated. We can indicate the acceptable hyphenation points by inserting "\-" as follows:

```
ar\-ti\-cles
```

If the justification is improved by it, TEXT will hyphenate the word at one of the suggested points, ignoring any "\-" it does not use. If a word contains a hyphen already (e.g. stand-alone), TEXT will treat that hyphen as a primary hyphenation point. Since it is ludicrous to think of typing hyphenation points for every word you enter, these instructions are left for insertion at the polishing phase of our review, and are added only to words that begin a line following a short (filled) or thin (justified) line.

If you have designated hyphenation points that you later decide you do not want used, or if you have a hyphenated word, such as RS-232, that you do not want broken at the hyphen, you may turn off the hyphenation feature with the instruction:

```
.nh
```

No "\-" will appear in the output, and no words will be hyphenated To return to hyphenation later in the same document, use

```
.hy
```

to turn the hyphenation feature back on. TEXT runs initially with the hyphenation mode turned on.

Edit the sample.fle now and add hyphenation points to the word "articles" in the second paragraph.

Run TEXT and store the formatted version of sample.fle in review.fle (text sample.fle review.fle). Check to see if TEXT used any of the specified hyphenation points.

NON-PRINTING CHARACTER

What if you need to begin an input line with a period (.)? TEXT always assumes that such a line is an instruction line and will try to process it as such, throwing away everything on the line it does not understand. If you must begin a text line with a period, perhaps because you are writing an article about TEXT's ".rf" instruction, you can use the non-printing character "\&". This character has no width and does not affect word or line spacing, but TEXT recognizes it as a character so any line it begins will be treated as a text line.

\&.rf is one of the most versatile

PRINTING THE FINAL DOCUMENT

Once you have finished writing your document and adding the TEXT instructions you want, it is time to print out the formatted document on your printer. TEXT provides several flags and instructions to make this process easy and productive.

TEXT Flags

If you have a continuous roll of paper mounted on your printer, you may have TEXT supply a cutting line between each page so that you will know exactly where to separate the pages to keep your header and footer space consistent on every page. To invoke this feature, use the "-c" flag when you run TEXT:

```
text -c sample.fle lp:          HDOS or
text -c sample.fle lpt:         CP/M
```

TEXT will then print

---

between each page.

If you are using single sheets of paper, rather than the normal fanfold computer paper, you may have TEXT stop after each page to allow you to insert a new sheet. When you use the "-w" flag with the TEXT command, TEXT will print a page and then signal with a beep that it is waiting for a RETURN from the keyboard to continue. If you have letterhead paper, for instance, that you want to use for the first page of your letter, insert your letterhead in your printer and type the command

text -w sample.fle lp:

TEXT will beep to ask if you are ready to have the first page printed. Type a RETURN from your terminal keyboard. After printing the first page, TEXT will stop and beep again to ask if you are ready for the second page. Insert one of your plain second sheets in the printer and type RETURN when you are ready. TEXT will then

print the second page. If there are additional pages, TEXT will stop after each one to allow you to insert a new piece of paper.

Suppose that as you are watching TEXT print out your final copy you notice that you forgot to add a ".fi" instruction after the table on page 12 and everything after that is formatting in no-fill mode. You abort your command with a "CTRL-C" and add the missing instruction. The first 11 pages of your document are printed just fine so you would like to begin printing with page 12. TEXT allows you to specify this with the "-o" flag. By typing

```
text -o12 sample.fle lp:
```

you may have TEXT begin output with page 12.

You then proofread your final copy one last time and discover a small mistake on page 4 that will require reprinting that page. After making the correction in your document file you may have TEXT print only page 4 by using the "-o" and "-w" flags together. When you type

```
text -w -o4 sample.fle lp:
```

TEXT will print page 4 and then beep and wait for a RETURN from the keyboard before going on to page 5. If you type a "CTRL-C" to abort the program instead, you will have printed only page 4.

IN-LINE PAUSE

TEXT has an instruction which allows you to specify a "wait" at other places than just between pages during the printing of your document. If you have a printer with changeable daisy wheels (or type balls), for instance, TEXT will allow you to change these as often as you want. You simply insert the in-line wait instruction "\w" in your document file before each daisy wheel change. When TEXT encounters the instruction, it will stop printing, beep and wait for a RETURN from the terminal keyboard before continuing. You may then change the daisy wheel and type RETURN. For example, when TEXT formats the sentence:

```
We offer \wFIVE\w varieties from which to choose
```

it will stop at each "\w" so that the word "FIVE" may be printed in a different typeface.

[NOTE: If you have a single output line that uses more than one typeface, in order to get proper underlining, emboldening or overstriking with TEXT, your printer must have character backspace capability and you must be in backspace mode (see the instruction ".bs 1" on page 26 for more information) during output of the line.]

TERMINAL MESSAGES

It is a good idea to have TEXT print a message on your terminal to remind you of the reason for a particular \w pause. The ".tm" terminal message instruction provides this capability. When it is encountered by TEXT, everything that follows it on the instruction line is displayed on the terminal screen. The message does not appear in nor in any way affect the formatting of your document file. For instance, if you precede the above sentence with

```
.tm Change to italic and back again.
We offer \wFIVE\w varieties from which to choose
```

Your screen will look like this after TEXT prints "We offer ", beeps and stops:

```
>text sample.fle lp:
Change to italic and back again.
```

or

```
A>text sample.fle lpt:
Change to italic and back again.
```

This way, even if you run the document some time after you first inserted the "\w" instructions, you will know that you need to mount your italic daisy wheel before typing RETURN at the first wait, and remounting your regular daisy wheel at the second wait.

If your document has information that must be updated periodically, you can include a ".tm" and "\w" instruction at the beginning of your document file as a reminder to check this information before printing.

```
.tm The data in this file was last updated 3/25/82.
.tm If it is current, type RETURN to begin output.
\w
```

PRINTER CONTROL CHARACTERS

To include special printer instructions (known as printer control characters) in your document file, you must enclose them in the zero-width in-line TEXT instructions, "\(" and "\)". These instructions tell TEXT that all characters between the two are to be passed directly to your printer without processing and without being considered a part of the line for filling or justification purposes. The "\(" signals the beginning of the zero-width characters and the "\)" ends them.

For instance, to have your Diablo printer move your paper half a line down so that you can print a superscript, and then move it back again you would enclose the printer instructions inside the zero-width instructions

```
This is the fifth time\(ESC-D\)1\(ESC-U\) these
results were obtained from this experiment
```

TEXT will justify the line as if the zero width instructions and their contents were not there and the number 1 will print half a line up.

COMMAND LINE INSTRUCTIONS

It is possible to/ specify a limited number of separate line instructions on the TEXT command line. Instruction with arguments must be enclosed in double quotes. For instance, if you would like get a printed listing of one of your programs, you can have TEXT divide it into numbered pages without inserting a single TEXT instruction in the program file:

```
text .nf ".he xx-48-xx" program.fle lp:
```

TEXT will process the entire program.fle in no-fill mode (so default margin settings are ignored) and will give a centered page number at the top of every page.

Instructions given on the command line are processed BEFORE the first line of the named file. If you specify no-fill mode on the command line, but have a specific fill ".fi" instruction at the top of your file, your output will be filled because the fill instruction was processed after the no-fill instruction.

Run TEXT on our sample.fle and specify double spacing from the command line. Use the -w flag and review the output on the screen.

```
text -w ".ls 2" sample
```

FINAL EXAM

Run TEXT on sample.fle and store the output in sample.txt.

You have now completed this TEXT tutorial. Your sample.fle should look very much like the **sample.in** supplied on your distribution disk (although some of the instructions may be in a different order). Your sample.txt file should look just like the **sample.out** file provided.

Run TEXT on sample.fle again and print the result. Now run TEXT on the sample.in file and print it also. Compare the two final results. If there are any differences, study both input files and make sure you understand where those differences are coming from.

The instructions and features described in this tutorial are indexed at the back of this manual for your future reference. In addition, Sections 3 and 4 provide quick reference summaries of the TEXT instructions and commands available to you.

3. SUMMARY OF FORMATTING INSTRUCTIONS

TEXT formats documents by processing instructions which are embedded in the document input file.

There are two types of TEXT instructions. The separate line instruction consists of a period (.) followed by a two-character name, possibly followed by one or more arguments. Each instruction must be placed on a line by itself in the input file with the period in column 1. Any line beginning with a period will be assumed to be a TEXT instruction and will not appear in the output.

The in-line TEXT instruction consists of a backslash (\) followed by a single character (two characters for string substitutions). Each of these in-line instructions may appear anywhere in your input file, embedded in a text line or on a line by itself. If one is placed on a line by itself, that line is treated as a normal text line, unlike a line occupied by a separate line instruction which is disregarded during formatting.

SEPARATE LINE INSTRUCTIONS

.bd	embolden characters on following input line
.bd n	embolden characters on following n input lines
.bd 0	discontinue emboldening
.bn n	overstrike emboldened characters n times
.bp	begin a new page
.bp n	begin a new page numbered n
.bp +n	begin a new page and skip n page numbers
.br	begin a new output line
.bs 1	use character backspace for overstrike & underline
.bs 0	use multiple passes on line for overstrike & underline
.ce	center following line between current margins
.ce n	center following n lines between current margins
.ce 0	discontinue centering
.cu	continuously underline characters on following line
.cu n	continuously underline characters on following n lines
.cu 0	end continuous underlining
.fl n	set space from last line of text to footer to n lines
.f2 n	set space from and including footer to bottom of page to n lines
.fl +n	increase space from last line of text to footer to n lines
.f2 +n	increase space from and including footer to bottom of page by n lines
.fl -n	decrease space from last line of text to footer by n lines
.f2 -n	decrease space from and including footer to bottom of page by n lines
.fi	begin filling text (fill mode)
.fo "str"str"str"	specifies three-part footer

.h1 n	set space from top of page to and including header to n lines
.h2 n	set space from header to top of text to n lines
.h1 +n	increase space from top of page to and including header by n lines
.h2 +n	increase space from header to top of text by n lines
.h2 -n	decrease space from header and top of text by n lines
.he "str"str"str"	specifies three-part header
.hi string n	hanging indent string begins at character position n
.hi string -n	hanging indent string begins n spaces to left of current left margin
.hy	hyphenate at specified hyphenation points
.in n	set left margin to n
.in +n	move left margin n spaces to right
.in -n	move left margin n spaces to left
.ju	right justify following text
.ls n	set line spacing to n
.ne n	need n more lines on page, else begin new page
.nf	do not fill following text (no-fill mode)
.nh	do not hyphenate
.nj	do not right justify following text
.pl n	set page length to n lines
.pl +n	increase page length by n lines
.pl -n	decrease page length by n lines
.pp	indent following line 5 spaces; need 2 more lines on page else begin new page
.rf filename	read in specified file
.rm n	set right margin to n
.rm +n	move right margin setting to right n spaces
.rm -n	move right margin setting to left n spaces
.sb x string	substitute string for each occurrence of \sx
.sp n	space down n output lines
.tc "str"x"str"	three-part table of contents entry
.ti n	set left margin to n for next line only
.ti +n	move left margin to right by n spaces, one line only
.ti -n	move left margin to left by n spaces, one line only
.tm message	print message on screen
.uc	return underline character to underscore
.uc x	change underline character to x
.ul	underline non-blank characters on following line
.ul n	underline non-blank characters on following n lines
.ul 0	end word underlining
.we	discontinue writing to file specified in preceding .wf
.wf filename	write following lines in filename
.\"	ignore this line

IN-LINE INSTRUCTIONS

\<	begin emboldening
\>	discontinue emboldening
\{	begin word underlining
\}	discontinue word underlining
\[	begin continuous underlining
\]	discontinue continuous underlining
\(	begin zero width characters
\)	end of zero width characters
\-	acceptable hyphenation point
\#	substitute current page number
\"	comment; ignore everything to the right on the line
\(space)	unpaddable space
\&	non-printing character
\\	print \
\b	backspace one character position
\sx	substitute the contents of string x defined in previous .sb
\w	wait for RETURN from terminal keyboard before continuing

In-line instructions other than \# and \sx included between ".wf" and ".we" instructions must be double backslashed (\\) for proper processing.

4. THE TEXT COMMAND

RUNNING TEXT

To have TEXT format the contents of a file, type the TEXT command at your operating system prompt:

```
>text filename                    (HDOS)
```

or

```
A>text filename                   (CP/M)
```

TEXT will display the formatted output on your screen.

To have TEXT store the formatted output in another file for review, add the name of that file to the command

```
text filename review.fle
```

To have TEXT print your formatted document on your printer, substitute the name of your printer device driver for the review file name:

```
text filename lp:                 (HDOS)
```

or

```
filename lpt:                     (CP/M)
```

AVAILABLE FLAGS

TEXT offers several output options which may be specified on the command line between the program name and the file name. Multiple flags may be specified in any order.

```
text -t filename
text -c -w filename
```

To have TEXT supply a cutting line between each page for continuous roll paper, use the "-c" flag

```
text -c filename lp[t]:
```

To have TEXT skip to a specified page to begin output of the remainder of the formatted document, use the "-o" flag with the page number

```
text -o32 filename lp[t]:
```

To have TEXT disable the H19/Z19 inverse video display feature for emboldened and underlined text, use the "-t" flag

```
-t filename
```

To have TEXT wait for a RETURN from the terminal keyboard before formatting each page, use the "-w" flag

```
text -w filename lp[t]:
```

To have TEXT display its version number, use the "-v" flag:

```
text -v
```

TEXT will respond with a line like:

Text 4.0, 28 Mar 82. (c) 1982 J. J. Gillogly. All rights reserved.

TEXT INSTRUCTIONS ON THE COMMAND LINE

Any separate line TEXT instruction (except the substitute string ".sb" instruction) may be invoked from the command line by enclosing it in quotes and placing it after the program name. It will be processed as if it were inserted at the top of the specified file. Multiple instructions may be specified in any order, but must each be enclosed in quotes.

```
text -t ".pl 24" filename
text -w .nf ".bp 20" -o22 filename
```

SUMMARY OF TEXT COMMANDS

```
text filename
text filename review.fle
text filename lp[t]:
text -flag -flag filename
text -flag -flag .xx ".zz n" filename
```

SUMMARY OF TEXT FLAGS

-c	cutmarks
-o*n*	begin output with page *n*
-t	disable H19/Z19 inverse video for bold and underline when displayed on the screen
-v	display the version number of this TEXT program
-w	wait for a RETURN from the terminal keyboard before each page is formatted

5. SPECIAL USES: FORM LETTERS, TABLE OF CONTENTS & INDEX, FOOTNOTES

In addition to the standard text formatting functions, TEXT can be used for several special applications. These include generating form letters, creating a table of contents, gathering and formatting end-of chapter footnotes, and, with the help of a sort program, even creating an index.

FORM LETTERS

If you have a form letter you want addressed to a list of people, you can use the read file ".rf" and string substitution ".sb" instructions in TEXT to produce the letters automatically.

Figure 5.2 is an example of a form letter input file, here called **form.ltr**, which will be read in repetitively from the master mailing list input file shown in Figure 5.1. The file **letter.hd** referred to on the first line of form.ltr contains the instructions and text for formatting the return address at the top of each page. The file **initlize.txt** read in at the top of the master file contains the TEXT instructions that set the basic format of the letter such as margins, header, and footer.

==========

Figure 5.1: the mailing list used as the master input file for generating multiple copies of form.ltr, each addressed to a different person on the list

```
.rf initlize.txt
.sb z 28 March 1982
.sb a Mr. Timothy Russell
.sb b Electronics Today
.sb c 1366 Belmont Avenue
.sb d Vancouver, B.C
.sb e Canada
.sb f Mr. Russell
.sb g 1500.00
.rf form.ltr
.sb a
.sb b
.sb c Ms. Helen Boyce
.sb d 20937 W. Clandon Rd.
.sb e Troy, MI  48084
.sb f Ms. Boyce
.sb g 1200.00
.rf form.ltr
.sb a
.sb b Mr. Charles E. Ruppert
.sb c ADS Insurance
.sb d Box 2258
.sb e New York, NY  10010
.sb f Mr. Ruppert
.sb g 2000.00
.rf form.ltr
```

==========

Figure 5.2: input file form.ltr
of form letter and envelope to be used in conjunction
with a mailing list

```
.rf letter.hd

                                    \"date


.nf
\sa                                 \"line 1 of address
\sb                                 \"line 2 of address
\sc                                 \"line 3 of address
\sd                                 \"line 4 of address
\se                                 \"line 5 of address

     \sf

.fi
We here at Beneficial Investments are very excited about the
new service our
company is offering to established customers.  We are now able to offer
you 30-day interest-free credit on orders of $100.00 or more.
Your credit limit has been extended to $\sg.



We hope to hear from you soon

.nf
Sincerely,



K. K. Kruger
Credit Manager
.bp
.tm Insert envelope in printer and type RETURN to continue.
\w
.in +20
\sa                                 \"line 1 of address
\sb                                 \"line 2 of address
\sc                                 \"line 3 of address
\sd                                 \"line 4 of address
\se                                 \"line 5 of address
.in -20
.bp
.tm Insert sheet of stationery in printer and type RETURN to continue.
\w
```

If you prefer to do all your envelopes at once, you can remove the envelope instructions from the form.ltr file and put them in a file of their own. After you have run all your form letters, rename the envelope input file to form.ltr and run the mailing list again to address an envelope for each form letter you produced.

TABLE OF CONTENTS

A table of contents can be generated automatically with TEXT by using the write file instructions (.wf and .we) and the table of contents entry instruction (.tc). These instructions are covered in the tutorial section of this manual, under "Writing Temporary Files During Formatting."

When you create the table of contents file with the first ".wf" instruction, you should input all the TEXT instructions for the page title and general format. For instance:

```
.wf tablec
.bp 3                \"number the page 3
.fo ""-\\#-""
.ce
.cu
.bd
TABLE OF CONTENTS
.sp 3
.we
```

The page number reference (\#) must be double-backslashed so that it is processed when the tablec file is formatted. If it were entered with a single backslash, TEXT would insert the current page number into the footer instruction before writing it in the tablec file.

Then, each time you begin a subject in your document that you want to include in your table of contents, you make an entry to your table of contents file using the ".tc" instruction. For example,

```
.ce
Chapter One: \<The Nile Valley\>
.wf tablec
.tc "The Nile Valley "." \#"
.we
```

After TEXT has centered and emboldened the chapter title, it will write the ".tc" instruction line at the end of your tablec file, substituting the current page number for the "\#". Ten pages later when you begin Chapter Two you again make a table of contents entry:

```
.ce
Chapter Two: \<The Pyramids\>
.wf tablec
.tc "The Pyramids "." \#"
.we
```

If you have a section heading you also want to include in the table of contents you could add instructions to have it indented:

```
.bd
Valley of The Kings
.wf tablec
.ti +3
.tc "Valley of The Kings "."
.we
```

When you run TEXT the table of contents file will look like this:

```
.bp 3
.fo ""-\\#-""
.ce
.cu
.bd
TABLE OF CONTENTS
.sp 3
.tc "The Nile Valley "." 5"
.tc "The Pyramids "." 14"
.ti +3
.tc "Valley of The Kings "." 16"
```

At the end of your document, after all the table of contents entries have been made, you can have TEXT format the tablec file with the instruction:

```
rf tablec
```

and TEXT will produce

TABLE OF CONTENTS

with "-3-" centered at the bottom of the page.

INDEX

If you want your document to have an index, you insert the index entries just as you did with the table of contents entries, having TEXT write them to a file with the ".wf" and ".we" instructions.

For instance, after the table of contents entry for Chapter Two above you might add

```
.wf index
.tc "pyramids "." \#"
.tc "burial "." \#"
.tc "King Tut "."\#"
.we
```

You would add similar instructions at each place in the document where there is something you want indexed. When you run TEXT on your file, all of the index entries and corresponding page numbers will be stored in the named file. You will then need to sort the file alphabetically and throw away all duplications. For instance, the index file in our current example might include the following lines when sorted:

```
.tc "pyramids "." 15"
.tc "pyramids "." 18"
.tc "pyramids "." 26"
.tc "Ra "." 17"
```

The next step is to combine the separate page number references each entry.

```
.tc "pyramids "." 15, 18, 26"
.tc "Ra "." 17"
```

The last step before printing, is to add any TEXT instructions you want to format the page and title.

When you run TEXT on the file, the lines in our example will like this:

```
pyramids ..... 15, 18, 26
Ra ..... 17
```

FOOTNOTES

End-of-chapter or end-of-document footnotes are also formatted using the write file instructions. As you are typing your document, you may enter each related footnote and its formatting instructions in a write file. For instance,

```
Statistical analysis over a three year period
reveals a significant decrease
in the life expectancy of these rats.[3]
.wf footnote
.in +5
.hi [3] -5
These figures are based on the extensive laboratory
work of Dr. Hans Goodwin as reported in the
Spring 1980 issue of the
\{Scientific Journal of Fieldhard University\},
Prairie Flats, WI.
.we
```

Then, at the end of your chapter or book, you may have TEXT insert the footnote file by using the read file ".rf" instruction to have the footnotes included at that point in the document.

6. TROUBLE-SHOOTING

WHEN SOMETHING GOES WRONG

You have prepared your document for formatting and printing with numerous TEXT instructions inserted in your document input file. Everything looks like it should run beautifully, but it doesn't. What do you do now?

First try to isolate the problem area. Run TEXT and store the output in a file.

1. If it works perfectly when output to your screen or a file, but does not format properly when you try to print it, the problem is either with your printer or your device driver.

   If you are getting strange page breaks and you are running under HDOS, your device driver may be trying to set the page length and interferring with that set by TEXT. You can solve this problem with the simple command:

   ```
   >set lp: page 0
   ```

   Depending on the type of printer you are using and the device driver (file.dvd) you are using with it, you may have any number of problems with interactions between them and TEXT. In some instances, you will want to use the printer function instead of the TEXT function, using the zero-width instructions "\(" and "\)" to set off the printer control characters in your document file. In other instances, such as the one above, you will want to set your device driver to allow a TEXT function full rein.

2. If your file does not format properly in your review file, first double-check your formatting instructions. There is an index provided with this manual to help you check for the proper usage of any TEXT instructions. Chances are TEXT is just doing exactly what you told it to do, though it wasn't what you meant for it to do.

   One tricky problem we have encountered in the past is generated by a TEXT instruction that is used to manipulate the footer or header areas of a page to give a better page break and that is not returned to normal. For instance, if you have a ".fl +1" instruction in a ".rf" read file that is read in several times and the initial value is not restored, you may get "footer creep" where the body of your text slowly shrinks during formatting as the footer space size increases.

3. If all your instructions are correct, you should next check your **text.abs** or **text.com**. Do a checksum (CHECK available on the Utility Disk) or compare (CMP available on the C/80 program disk, both disks available through The Software Toolworks) on the text.abs you are using with the text.abs on your distribution disk. If you do not have either of these programs, make a fresh copy of text.abs or text.com on a new disk and run your file again.

4. If the text.abs (text.com on CP/M) checks out and you are still having problems, run a memory test on your machine to make sure it is not a bad memory chip.

5. If you still have not solved the problem, you may have discovered a TEXT bug. To pursue this possibility, you should first produce a minimal example. First, make a copy of your input file and remove all the text and TEXT instructions that appear AFTER the bug first occurs. Verify that the problem still exists. Next, make a copy of this pruned file and begin removing things from it that probably do not affect the problem area. If your header is misbehaving, for instance, you can probably take out all the emboldening and underlining commands within the body of the text as well as some of the text itself without disturbing changing the result. Remove a few things at a time, keeping a copy of the previous version each time, verify after each deletion that the problem still occurs. Eventually the problem will disappear and you will know that the most recent removals is the place to look for the problem.

   If the problem is indeed a TEXT bug, please send us a bug report consisting of the following:

   1. A copy of the minimal example input file that produces the bug.
   2. A copy of the output file produced when TEXT is run on this file.
   3. A description of the problem.
   4. The version number of TEXT (obtained by typing "text -v" at the operating system prompt), and the operating system you are running under.
   5. Your name, address and phone number if you would like us to get back to you after we have examined the problem.

   This information may be in listing form, or preferably, on a disk. If you are sending a disk, please also include à copy of the text.abs or text.com you used to generate item 2. above.

7. CHANGING THE DEFAULTS: PATCHES

DEFAULTS

The TEXT program supplied on your distribution disk is iniialized to the following values:

.pl 66	Page length 66 lines
.in 0	Left margin set to 0
.rm 60	Right margin set to 60
.hl 3	Number of lines from top of page to and including header, if any
.h2 2	Number of lines from .hl to first line of text
.fl 2	Number of lines from bottom line of text to footer
.f2 3	Number of lines from bottom of page to and including footer, if any
.fi	Lines of text are filled
.ju	Lines of text are right-justified
.hy	Words are hyphenated at hyphens and specified hyphenation points
.bn 1	Overstrikes are overstruck one time
.bs 0	Uses multiple passes over line for overstrikes and underlining
.uc	Underline character is the underscore
.ls 1	Single spacing between lines
(-t)	Overstrikes and bold are displayed on the screen in inverse video

All of these defaults may be overridden during processing with the use of the appropriate TEXT instruction. However, if you find you are **always** changing a particular one to the same setting, you might want to consider changing the default so that TEXT initializes each time to that setting. If, for instance, your printer has character backspacing but does not have the multi-pass capability assumed by TEXT for dealing with overstrikes and underlining, you could change the default so that TEXT automatically processes your documents as if you started every file with a ".bs 1" instruction.

In addition to changing the above defaults, you can change TEXT so that

it always prints cutmarks between pages
it always waits for a RETURN from the terminal keyboard between pages

KEYBOARDS WITH FOREIGN CHARACTER SETS

TEXT uses the backslash (\) as an escape character to introduce in-line instructions. It uses the pound sign (#) with the escape to represent the current page number. These characters were good choices for the English keyboard because they occur naturally in very few documents. However, if your terminal keyboard has been modified for a foreign character set, these keys may be needed for important letters and you will want to patch TEXT to use other characters for the escape and page number. In addition, you may want to specify up to six "dead keys". A "dead key" is a character,

usually an accent, followed by a backspace to give no apparent motion along the line

HOW TO PATCH TEXT

The file **text.pch** on your distribution disk contains a list of the necessary patch addresses and initial values you need to make changes to the TEXT defaults. This subsection gives directions for making the patches to TEXT using the patch program which comes with your operating system. We strongly recommend that you **do not** try patching your TEXT distribution disk. Make a copy first and patch the copy.

On HDOS: The PATCH program is included on the HDOS system disk. To patch TEXT, copy PATCH.ABS onto the system disk in use, if necessary. Then type the command PATCH. When the program asks for a file name, type TEXT.ABS. PATCH will then ask for the address; type the address you wish to change. PATCH will type the present value. To insert a new value, type the value and a return; to leave the present value unchanged just type a return. PATCH will go on to the next location. Typing a CTRL-D returns to the address prompt; typing another CTRL-D closes the file; a third CTRL-D exits to the HDOS monitor.

On CP/M: Make sure the files DDT.COM (included in your CP/M distribution) and TEXT.COM are on your current working disk. Then type the command "DDT TEXT.COM". When the "-" prompt appears, you may type the command "Snnnn", where **nnnn** is the hex address to be patched. The address and contents are displayed. You may type a hex value followed by RETURN to insert that new value, or a RETURN to leave the old value unchanged. Typing the character "." followed by RETURN will return to the "-" prompt. When you have made all desired patches, type CTRL-C to exit. When the "A>" prompt appears, type the command "SAVE 85 TEXT.COM".

INDEX