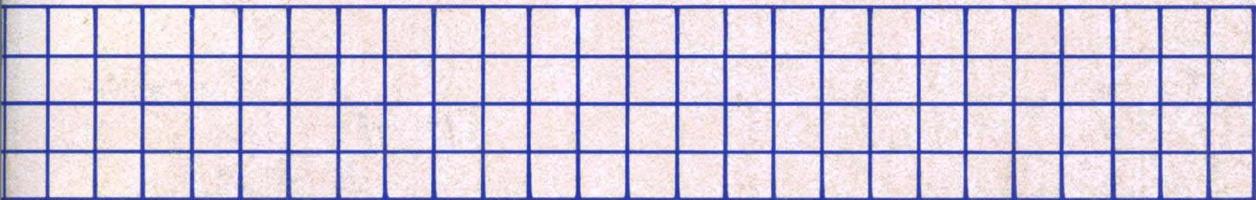


Perfect Writer  
Perfect Speller



**PERFECT WRITER™**

### **COPYRIGHT**

Copyright, 1982 by Perfect Software, Inc. All rights reserved worldwide. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system or translated into any human or computer language in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the express written permission of Perfect Software, Inc., 1400 Shattuck Avenue, Berkeley, California 94709.

### **DISCLAIMER OF WARRANTY**

Perfect Software, Inc. makes no representations or warranties, either express or implied, with respect to this manual and accompanying software and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. This manual and accompanying software are sold "as is" and Perfect Software will in no event be liable for direct, indirect, incidental or consequential damages resulting from any defect, error or failure to perform.

### **TRADEMARK**

Perfect Writer<sup>TM</sup>, Perfect Speller<sup>TM</sup>, Perfect Mailer<sup>TM</sup>, Perfect Sort<sup>TM</sup>, Perfect Terminal<sup>TM</sup>, Perfect Messen  
ger<sup>TM</sup>, Perfect Calc<sup>TM</sup>, Perfect Ledger<sup>TM</sup>, Perfect Software<sup>TM</sup>, and the Perfect<sup>TM</sup> prefix are trademarks of Pe  
fect Software, Inc.

---

# **PERFECT WRITER USER'S GUIDE**

## I. Introduction

### **PART I WORD PROCESSING FUNDAMENTALS**

- II. Getting Acquainted
- III. Beginning and Quitting
- IV. Moving the Cursor
- V. Deleting and Inserting
- VI. Storing a Document
- VII. Printing a Document
- VIII. Modes and Additional Commands

### **PART II ADVANCED EDITING PROCEDURES**

- IX. Editing Multiple Files
- X. Split-Screen Editing
- XI. Copying and Moving Text
- XII. Searching

### **PART III DOCUMENT DESIGN**

- XIII. Overview
- XIV. Environment Format Commands
- XV. Typeface Format Commands
- XVI. Document Sectioning and Organization
- XVII. Tools for Form Letter Design
- XVIII. Document Style Commands
- XIX. Document Design Lessons

### **APPENDICES**

- A. Installation
  - B. Error Messages
  - C. Swapping . . .
  - D. Glossary
  - E. Index
-



## Chapter I INTRODUCTION

Welcome to the world of Perfect Writer, where computers are 'friendly' and where the writing, editing, formatting, and printing of written documents is quick and easy.

Perfect Writer is an advanced design software system that represents the latest available technology in word processing. Once installed in your computer it will allow you to perform a wide variety of text editing and printing tasks with an ease you never dreamed possible.

This User's Guide will introduce you to Perfect Writer. No doubt you view the reading of a large computer user's guide as a demanding requirement. However, have no fear, because the Perfect Writer User's Guide represents a radical departure from the the computer manuals you may be accustomed to. Its style and organization permit quick and easy understanding of every feature of the Perfect Writer system. Though it is comprehensive and complete, the User's Guide is nevertheless free of technical jargon and arcane 'computerese.' Numerous illustrations and diagrams accompany the structured, step-by-step explanations. In many instances, you will only have to glance at the illustration or example to grasp the principle involved. We are not exaggerating when we say that you will probably begin using most of the basic features of Perfect Writer within 25 to 30 minutes after putting it on your computer.

### **Organization of the User's Guide**

For your easy understanding and reference, the User's Guide is presented in three parts:

**Part One:** This section contains all the basic commands and instructions necessary to begin using Perfect Writer, including: moving the cursor, deleting and inserting text, storing, and printing documents. Where appropriate, illustrations and examples have been provided to help explain the material. Also, a number of exercises allow you to begin practicing what you learn.

---

Part Two: Here are presented those commands and procedures that involve details not briefly explained. These procedures include: multiple document handling, split-screen editing, searching, searching and replacing, and moving and copying text. Presented in the same straightforward manner, with many helpful examples, these procedures are quick and easy to learn.

Part Three: This part introduces you to Perfect Writer's document design and printing capabilities. Though by this time you will be thoroughly accustomed to the 'basic' printing options presented in the first section, the formatting options contained in this section are so simple and easy to learn, we think you will soon begin using them in almost every document you prepare.

Next, an Appendix contains Perfect Writer's installation instructions, a comprehensive glossary of terms, and a list of possible error messages that can occur.

Finally, a comprehensive index will help you quickly find whatever you want to know about Perfect Writer.

## **WHAT YOU GET WITH PERFECT WRITER**

### **STANDARD FEATURES**

- Straightforward steps for creating letters and documents.
- Control commands that move you quickly and easily throughout the document displayed on your screen.
- Simple, yet comprehensive, procedures for changing and editing a document.
- Safe and convenient routines for storing the material you create.
- A basic printing option that satisfies a wide variety of printing needs.
- Numerous internal safeguards that protect against irretrievable mistakes and accidental loss of your material.
- A large, flexible, and easily understood command language that is quick and easy to learn and that uses the standard keys of any computer console.

## **ADVANCED FEATURES**

### **Virtual Memory Architecture**

Virtual memory enables your computer to run programs which are larger than its internal memory. This is accomplished by an advanced software engineering design feature that swiftly and automatically transfers portions of core memory to and from disk storage. This exchange or 'swapping' allows text files to be processed that are larger than your computer's internal memory, thus permitting convenient editing of very large documents. Most amazing of all, the operation is so efficiently and automatically performed by Perfect Writer that one has almost no awareness of it occurring, except that working memory seems almost infinite!

### **Multiple File Buffers**

Perfect Writer is a 'multi-buffer in-memory word processor,' which simply means you can access several document files at one time, a feature that is virtually unknown to other word processors. By copying documents into separate 'memory buffers,' it is possible to switch back and forth between documents with ease. In fact, Perfect Writer allows simultaneous access and editing of up to seven documents at one time.

### **Multiple File Display**

In order to take full advantage of the multiple file buffers and virtual memory architecture, Perfect Writer provides a multiple file display that allows you to view, compare, and edit two documents simultaneously. Using a split screen, it is possible to gather parts of text from one file and insert them into another file while viewing the process on the screen. This flexibility is unmatched by any other available word processor.

---

## Sophisticated Document Design Capabilities

Perfect Writer includes a document design program which allows you to create a 'perfect' layout for your document. When preparing a document on a conventional typewriter it is necessary to format the text by continually aligning and realigning margins, tab spacings, line spacing, etc. Most word processors today require that you follow these same old fashioned procedures, except that now you are setting and resetting the margins on your computer instead of the typewriter.

Perfect Writer does away with all this. With Perfect Writer you simply indicate, using a single word or symbol, how the text is to be formatted. Will it be a quotation? A verse? A numbered list? A footnote? Will the heading be a chapter heading, a subheading, a section, an appendix? Knowing what kind of format you want for a particular portion of text, Perfect Writer automatically invokes a predefined format option (there are more than 30). Automatically, Perfect Writer centers the text, justifies it, underlines it, boldfaces it, italicizes it, indents it, numbers it, surrounds it with white space (in pleasing proportion), makes an entry for it in the table of contents—whatever! You don't have to concern yourself. Of course, you could do it all in the old fashioned way, if you want to...but why bother? For each of its standard formats Perfect Writer provides 'style' options that allow you to adjust the formats to your personal preferences if the default settings do not suit you.

## ADVANCED DOCUMENT DESIGN FEATURES

### Table of Contents

Perfect Writer automatically creates a table of contents, listing and numbering the chapters, sections, subsections, headings, paragraphs, and appendices included in your document. In addition, the table of contents produced includes the page numbers where the listed text begins.

### Index

Perfect Writer automatically creates an alphabetized index of all words and topics tagged in a text, indicating the page where the words or topics appear.

---

## **Footnotes**

Depending upon your preference, Perfect Writer will automatically place your footnotes at the bottom of the page, the end of the document, or within the body of the text itself. The footnotes are sequentially numbered, and if you add new footnotes, Perfect Writer automatically renames from the beginning.

## **In-Text Referencing**

Perfect Writer allows you to tag topics or items for later reference. For example, if you discuss 'Theory X' on pages 5 and 6, then wish to reference it 10 pages later, Perfect Writer will automatically determine on which page in the printed version the reference will occur, and it will insert this page number in your in-text reference. If you later revise the document, inserting or deleting pages and sections, Perfect Writer automatically adjusts the page numbering to keep your in-text references correct.

## **Form Letter Design**

Perfect Writer offers powerful and flexible options for creating and manipulating form letters.

- **Console Input:** Perfect Writer allows you to insert portions of text from the console while a document is being printed. This is especially useful for 'individualizing' form letters.
  - **Targeted Form Letters:** Perfect Writer allows you to print selected parts of a document. For instance, you can prepare a letter with a number of different closing paragraphs targeted to different groups. When the letter is printed you select the particular closing paragraph you want for each group.
-

## COMPLETE ADAPTABILITY

Most word processing programs are written in 'assembler language,' a low level, machine-oriented programming language. Programs written at this level do not make full use of the capabilities available in the microcomputers being built today. In contrast, Perfect Writer is written in 'C', a high level streamlined language which allows much greater flexibility and creativity in software design. More importantly, 'C' programs are highly 'transportable,' which means that although Perfect Writer was originally intended for the 8-bit Z-80 machines, the program will also operate perfectly well on the new 16-bit machines (such as IBM's Personal Computer), and even 32-bit machines. The next several years will see more and more personal computers appearing based on the 16- and 32-bit processors, because of the combined speed, accuracy, and memory size they offer. Compared to the current generation of 8-bit personal computers, these new machines represent a quantum leap in power and capability.

Constructed as they are, with outdated software technology and the primitive assembler language, current word processors will not be able to keep up with these major advances in computer hardware. Instead they will literally 'tie your computer in knots.' Perfect Writer on the other hand, because it is written in 'C', will not become obsolete, but will easily accompany advances in computer hardware through the year 2000, at least. This means that if you upgrade your computer hardware in the coming years, you can be safely assured that:

- Your text files will still be usable.
- You will not need to purchase a new word processor.
- You will not have to learn a new word processor.

## **FINAL WORD**

We are at the dawn of an historic revolution, equivalent in scope to the Industrial Revolution of the 19th century. In the Industrial Revolution, human beings were able to harness, through the application of scientific laws, vast new resources of power for industrial applications. In the personal computer revolution, human beings are again harnessing power, but of a different kind. It is the power of artificial memory and intelligence. As noted computer scientist Joseph Deken writes: "[Human beings] would find it useless to compete in raw muscle power with tractors and combines; it is equally futile to compete in raw memory power with a bank of video disks."

Through the decade of the 80's and beyond, personal computers will become increasingly important to us, not only in our businesses and professions, but in our personal lives. Their function will be one of freeing us from much of the numbing drudgery that has for so long hampered and stunted our intelligence and creativity. We sincerely hope that as you learn to use Perfect Writer, you will find this to be the promise of the new age: that your skills and abilities are increased and enlarged, and that Perfect Writer frees you to do more creative and productive work.

---

## Notes

---

## Part I

# WORD PROCESSING FUNDAMENTALS

### In This Section

---

- **GETTING ACQUAINTED**
  - How to type **COMMAND** keys
  - How to **BEGIN** and **QUIT** Perfect Writer
  - How to **ENTER** your text
  - How to **EDIT** your text, including:
    - \* Deleting
    - \* Inserting
  - How to **STORE** your text
  - How to **PRINT** your text
- 

This section of the User Manual contains all of the basic commands necessary to begin using the Perfect Writer word processing system. As you will see the material has been outlined and structured for quick and easy learning. Although items can be individually referenced, we recommend that you read the sections **sequentially** stopping to practice the exercises provided.

---



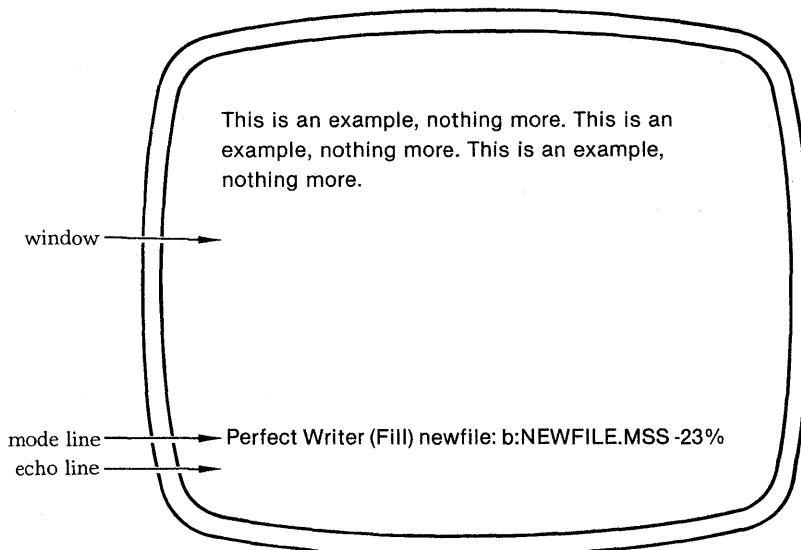
## Chapter II **GETTING ACQUAINTED**

This is where you should begin to learn about Perfect Writer if you have not had any experience with similar systems before. This chapter presents the major conceptual elements of a screen oriented editor like Perfect Writer, and introduces a number of terms and ideas that will be used later to explain how Perfect Writer works.

Because Perfect Writer shows you the text you are editing on the screen of your terminal, it is called a 'Screen Oriented Editor.' The Perfect Writer screen display is divided into three areas. The major portion is the 'window,' where the text of a document being edited is displayed. Two smaller portions, the 'Mode Line' and the 'Echo Line,' appear at the bottom of the screen, beneath the window.

## The Window

The window can hold 20 or so consecutive lines of a document at one time. The window always shows the current status of the text it contains. That is, as text is inserted or deleted, the screen reflects the change immediately. A fundamental principle of Perfect Writer is that what you see on the screen is what you actually have in your document.



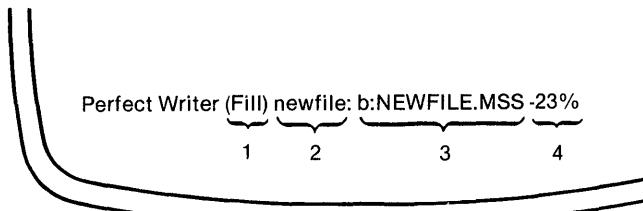
*Figure 1: Screen Display*

## The Cursor & Point

The screen display always contains within it the terminal's 'cursor,' a solid, sometimes blinking box or underline. The cursor is simply an indicator of the point where you are in your text.

## The Mode Line

The Mode Line is of special importance in the Perfect Writer screen display. It appears beneath the window and contains information relevant to the editing you are currently doing. A typical Mode Line might look like this:



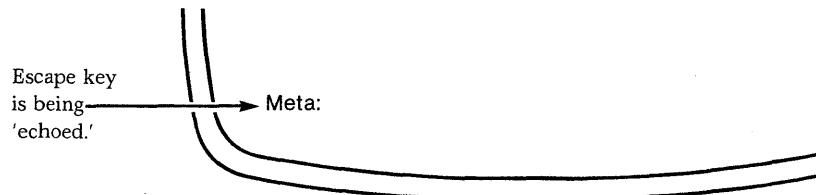
*Figure 2: Mode Line*

The elements of the Mode Line (here numbered) mean the following:

1. You are typing in the "fill" mode. In this mode, all words are wrapped to provide a constant right margin.
2. You are working in an editing buffer here called "Newfile," a name you would use when switching from buffer to buffer (see Chapter IX).
3. You are editing a file called "b:NEWFILE.MSS." The "b" indicates the disk drive where the file is located, in this case, the disk on drive "b."
4. The cursor is approximately 23% of the way through the file.

## The Echo Line

The third important area of the display screen is the line below the Mode Line, called the 'Echo Line.' The Echo Line is so-called because it "echoes" command prefix characters. For example, when the command key 'Escape' is typed, the message 'Meta:' will appear at the left of the Echo Line. The Echo Line is also used for displaying and accepting questions and additional information that Perfect Writer needs to complete certain commands. Finally, various system messages are displayed in the Echo Line.



## COMMAND KEYS

Perfect Writer uses two keys in combination with other characters to produce commands that will search, delete, store, and otherwise edit text material. These are:

- The **Control** key



- The **Escape** key†



In this section we will briefly discuss the operation of these two command keys in order to help you understand the material that follows. The emphasis will be upon the **mechanics** of typing commands, and therefore it will not be necessary to remember the specific commands given as examples. All commands will be discussed individually and in detail in the subsequent sections of this manual.

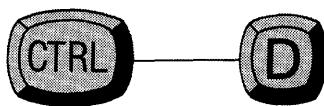
---

†Note: On some keyboards this key is labeled 'ESC,' 'ALT,' ALTMODE,' or 'META.'

## The Control Key

The Control key functions somewhat like a 'shift key' on an ordinary typewriter. Depressing and releasing it by itself produces no effect. However, depressing and holding it, while typing another character results in a command recognized by Perfect Writer.

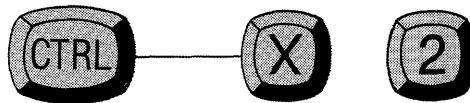
For example, depressing and holding the Control key while typing the letter 'D' tells Perfect Writer to delete a character.



(While holding the Control key, depress 'D')

**Note:** The solid line indicates that both keys are depressed at the same time.

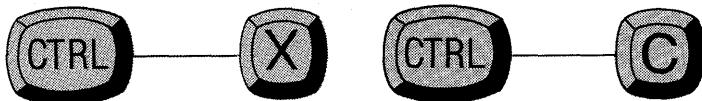
Sometimes the Control key is accompanied by two characters, as in the following CREATE TWO WINDOWS command. Here, the first character, 'X,' is depressed in conjunction with the Control key, while the second character, '2,' is not.



(While holding the Control key, depress 'X'; after  
releasing these, depress '2.')  

---

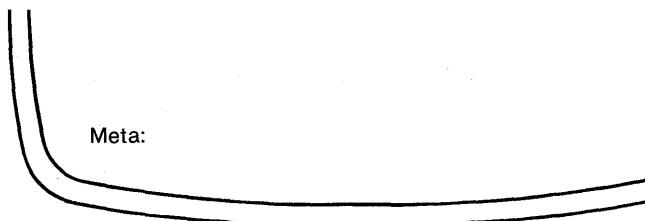
For some commands the Control key is used twice in succession, as in the following QUIT command, which tells Perfect Writer that you want to stop using the system:



(That is, while holding the Control key, depress 'X'; then, while holding the Control key **again**, depress 'C'.)

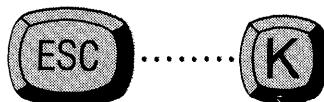
## The Escape Key

Like the Control key, the Escape key is also followed by a character in producing commands. The Escape key is used for an imaginary shift key like the Control key that we call Meta. The commands produced by the combination of the Control key and a character are referred to as (obviously enough) Control commands. The commands produced by the combination of the Escape key and a character are called Meta commands, as if they were produced by this imaginary shift key, the Meta key. However, unlike the Control key, the Escape key must not be held. It has only to be depressed and then released. Because of this it produces an effect that is lasting and cannot be ignored. To alert you that the Escape key has been depressed and that the character following it will result in a command, the following message is displayed in the Echo Line at the bottom of the screen:



This alerts you that the next character that you type will be interpreted as a Meta command. If you type the character that follows the Escape quickly enough, then this message will not appear.

**Example:** One Escape command is the DELETE SENTENCE command, which tells Perfect Writer to delete an entire sentence regardless of the number of lines it occupies. As with the Control key, no character 'K' will be printed on the screen:



(simply type the Escape key, followed by 'K')

**Note:** The dotted line indicates that the two keys need not be depressed at the same time.

### **Meta-Control Commands**

In a few cases the Escape key is used in combination with the Control key, as in the DELETE ENTIRE LINE command:



(Which means: first depress and release the Escape key;  
then, while holding the Control key, type 'K')

Here the Meta-Control command is simply an enhancement of an original Control command. That is, Control ----- K (DELETE LINE command) will delete all characters from the position of the cursor to the end of a line, whereas Escape... Control ----- K (DELETE ENTIRE LINE) will delete characters on both sides of the cursor.

In just this fashion Meta commands are often related to Control commands, performing a similar function but on a larger scale. This relationship makes the commands easy to learn and remember, and we will be pointing it out to you frequently in the rest of the manual.

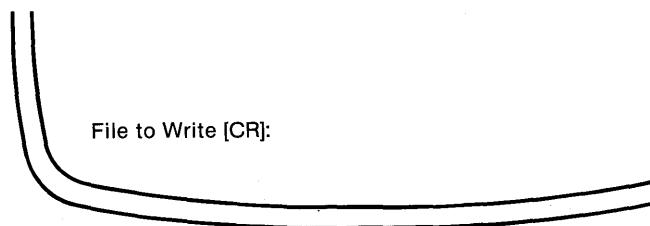
## No CARRIAGE RETURN

There is no need to type a RETURN ('ENTER' or 'SEND') to begin execution of a command. The display screen automatically and immediately reflects your command after the Escape or Control command characters have been typed.

## COMMAND MESSAGES

Sometimes Perfect Writer requires additional information regarding a command. In such cases, it will pause and ask for this information in the Echo Line at the bottom of the screen.

For example, at the command to WRITE FILE, Perfect Writer will ask in the Echo Line:



In this case, you must name the new file you have been editing, before Perfect Writer will write it on your disk for permanent storage.

**Note:** Every command that asks for further information will also indicate the appropriate system response to follow. In this example, it is a '[CR],' or 'carriage return.' After entering the file name type a 'carriage return' to tell Perfect Writer that execution of the command is now ready to continue.

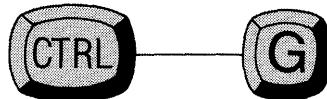
The Escape key is sometimes used in this way. When it is, '[ESC]' appears after Perfect Writer's question to you.

## **YES/NO Responses**

Sometimes in response to a command Perfect Writer will ask for a yes/no decision from you. Typing either 'Y' or 'N' is sufficient answer for Perfect Writer to complete the command.

## **The GO BACK Command**

Any command which Perfect Writer has **not yet begun executing** can be canceled by typing the GO BACK command. If you ever change your mind and want to stop or simply get stuck, just enter the GO BACK command:



## Notes

## Chapter III **BEGINNING & QUITTING**

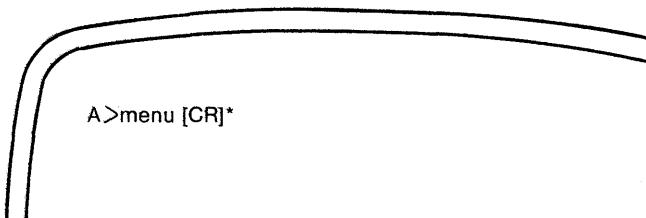
### **Beginning**

When you have writing, editing, and printing work to do, you will wish to begin the "Editing" Mode of Perfect Writer. To do this:

1. You will have to turn on your computer system and terminal, and call up DOS , the operating system. Usually this only involves turning the power on, inserting a diskette into the first drive, and depressing and releasing the 'Reset' button.
2. After you have entered DOS there will be a prompt that indicates that DOS is ready for you to give it a command.

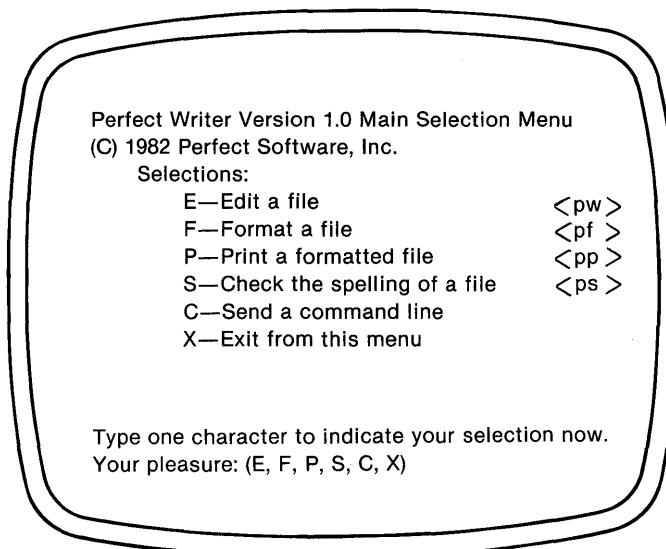


Type 'menu' and hit the carriage return. Enter:



\*Note: [CR] means hit return key.

This will call up Perfect Writer's Main Selection menu:



3. To begin an editing session, type the letter 'E' which will cause Perfect Writer to ask for the name of the file you wish to edit.
4. You may give a filename now, by typing the name followed by a carriage return or you may elect to assign a name later by typing only a carriage return.

After some disk activity you will be in Perfect Writer's editor.

If you would prefer not to use the menu system, you may merely type 'pw' followed by the filename of the file you wish to edit (e.g., "pw filename"). PW is the name of the Perfect Writer editor (see Chapter VII, page 6).

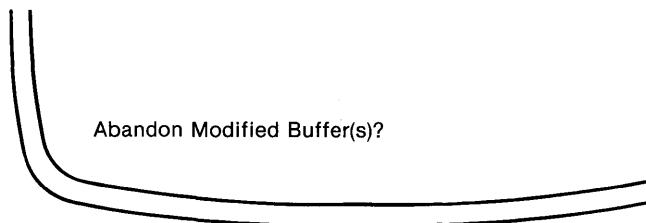
## **Quitting**

When you wish to stop working in the Perfect Writer editor, do this:

1. Type the QUIT command:



If you have edited or changed the document you have been working with, or if it is a new document, Perfect Writer will ask in the Echo Line:



2. Answer yes or no, by typing either "Y" or "N."

**Note:** The procedures for saving a new or edited document will be discussed later. Perfect Writer will always try to return you to the main menu level, if possible. If you entered from the menu then you will always be returned to the menu.

## ENTERING YOUR TEXT

This section explains how you can enter your text—a letter, an essay, a memorandum—into Perfect Writer.

### Steps:

1. Begin Perfect Writer as instructed earlier.
2. To enter text, begin typing as you would on a standard typewriter. As you type, your words are displayed in the window at the position of the cursor.
3. There is no need to use the 'carriage return' (or 'enter' key) to start new lines. Perfect Writer automatically begins a new line when necessary.
4. Use the 'Delete' key ('DEL' or 'RUBOUT') to correct typing errors. The Delete key erases the previous character and moves the cursor back one space.
5. Separate your paragraphs with blank lines, using the enter or return keys to create these. Paragraphs must be separated by at least one blank line.
6. Use the tab key to indent the first line of each paragraph, if you want.
7. As you near the bottom of the screen, your text will move up and redisplay, so that your cursor always stays visible.

**Note:** It is important to understand the action of the 'carriage return,' the 'tab' key, and the 'space bar,' all of which actually insert characters into the text. The characters are invisible but nevertheless quite real. They can be moved and deleted. They are quite different from the other 'blankspaces' on your screen.

The **space bar**, for example, inserts a character representing a single blank space. The tab key inserts a character representing eight character spaces.

The **return or enter key** inserts what is called a 'newline' character. Like the characters inserted by the tab and the space bar, it is invisible. Its sole function is to tell Perfect Writer to begin a new line at that point. Newline characters can be deleted just like any other character.

To see how the newline character functions, type several carriage returns in a row. Notice how the cursor is moved to a new line each time. Now type the Delete key several times. The cursor will be moved back to its previous line as each successive 'newline' character is deleted.

---

**Exercise:** Type the following document onto Perfect Writer, correcting typing errors as you go. Observe all indentations, separating paragraphs with single blank lines.

## **MEMORANDUM**

To: All staff

From: The boss

Re: New staff member

Perfect Writer has at last arrived, and you are all invited at your convenience to become familiar with it. I am speaking of our new word processor.

Though it arrived in a simple, brown paper mailer, Perfect Writer is really a beaut! And what a worker! Two weeks worth of paper piles disposed of in three days!

I know everyone will be crowding around Perfect Writer before long, so I have made arrangements that everyone gets a turn at the keyboard!

Of course, anyone who wants to be trained, will be. (Training doesn't take very long because Perfect Writer is probably the easiest word processor to learn.)

## Notes

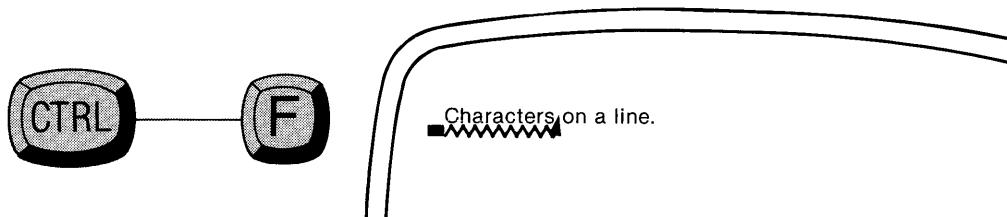
## Chapter IV **MOVING THE CURSOR**

You wish to begin changing and correcting a document that you have entered onto Perfect Writer. The first step is to position the cursor at the end of the word or line in the text to be changed.

The following commands move the cursor across lines and words of the text without deleting or changing any character. Note that most of the following cursor commands are mnemonically assigned; that is, 'F' for 'forward,' 'B' for 'backward,' etc.

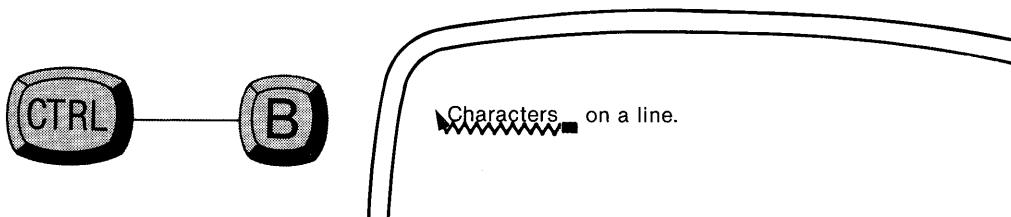
### **The FORWARD CHARACTER Command**

Moves the cursor one character forward along the line it occupies.



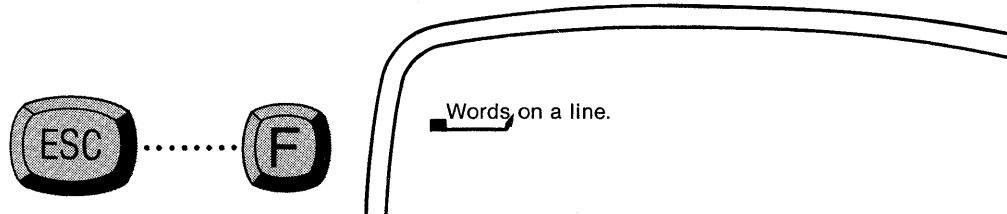
### **The BACKWARD CHARACTER Command**

Moves the cursor backward one character along the line it occupies.



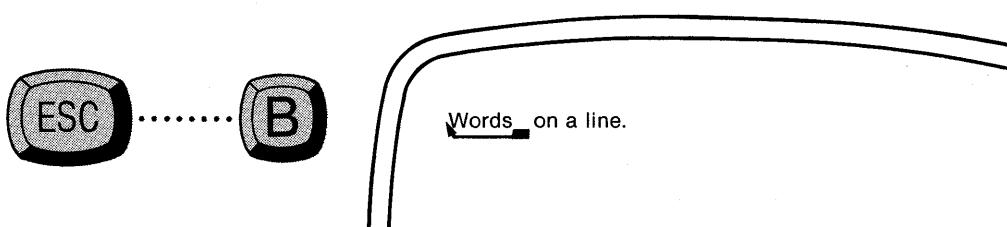
## The FORWARD WORD Command

Moves the cursor forward to the end of the word it currently occupies. If the cursor is not in a word, this command will move the cursor to the end of the next word.



## The BACKWARD WORD Command

Moves the cursor backward to the beginning of the word it is currently in. If the cursor is not in a word, this command will move it to the beginning of the previous word.

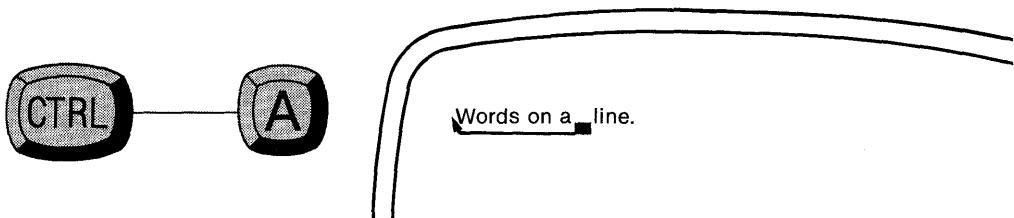


---

Note how Control ----- F and Control ----- B are related to the Meta commands Escape...F and Escape...B. The Control commands work on small units while the Meta commands perform similar editing functions only on larger units. This is part of the 'orthogonal command' feature used to make Perfect Writer easier to learn.

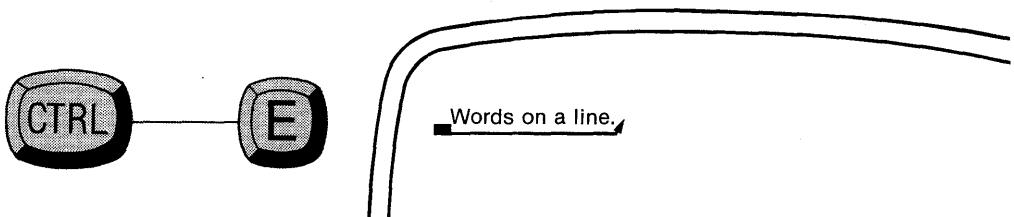
## The BEGINNING OF LINE Command

Moves the cursor to the beginning of the line it occupies. Remember this command by thinking of 'A' as being at the 'beginning' of the alphabet. Successive commands will move the cursor to the beginning of subsequent lines.



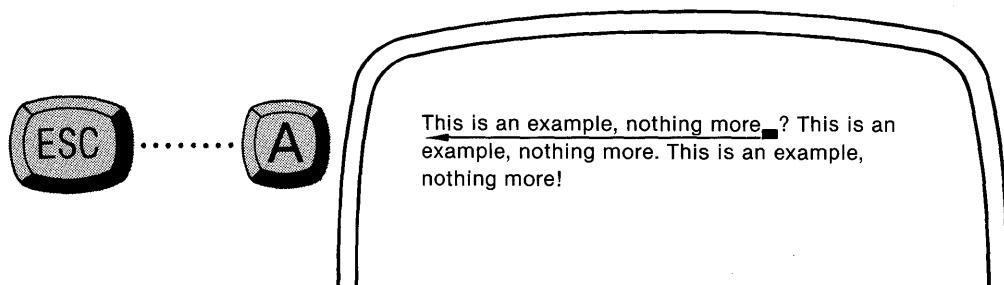
## The END OF LINE Command

Moves the cursor to the end of the line it occupies. Successive commands will move to the end of subsequent lines.



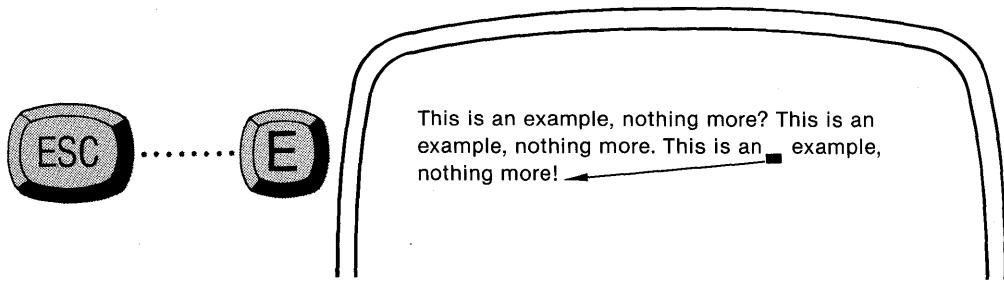
## The BEGINNING OF SENTENCE Command

Moves the cursor to the beginning of the sentence it occupies. If not in a sentence, the cursor moves to the beginning of the previous sentence.



## The END OF SENTENCE Command

Moves the cursor to the end of the sentence it occupies, i.e. until it encounters a period, '.', a question mark, '?', or an exclamation mark, '!'. If the cursor is not in a sentence, it will move to the end of the following sentence.

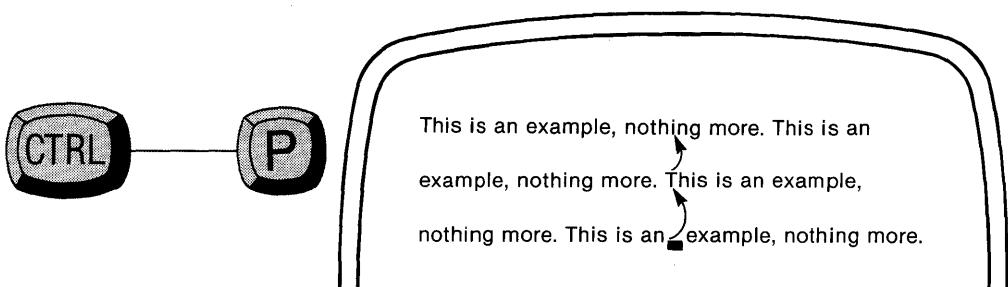


Note how Control—A and Control—E are related to the Meta commands Escape···A and Escape···E as mentioned earlier.

---

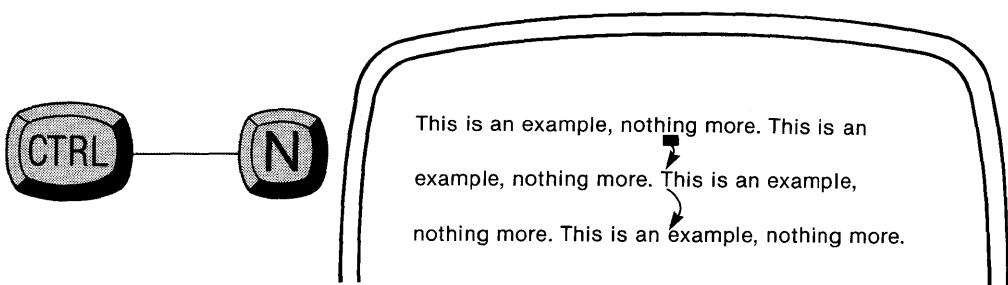
## The PREVIOUS LINE Command

Moves the cursor up to the previous line, keeping the cursor in roughly the same **column** as it moves.



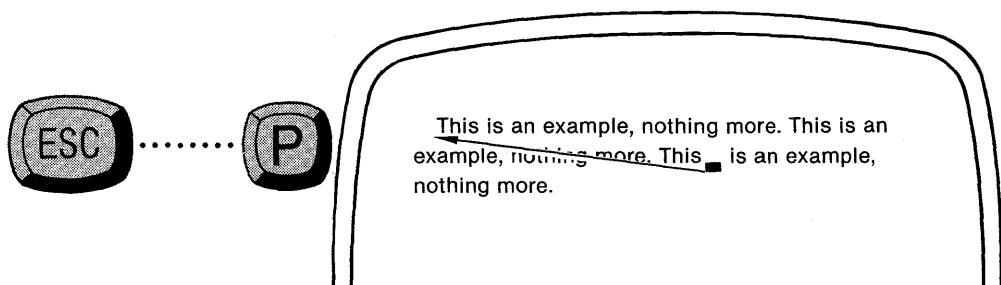
## The NEXT LINE Command

Moves the cursor down to the next line, keeping the cursor in roughly the same **column** as it moves.



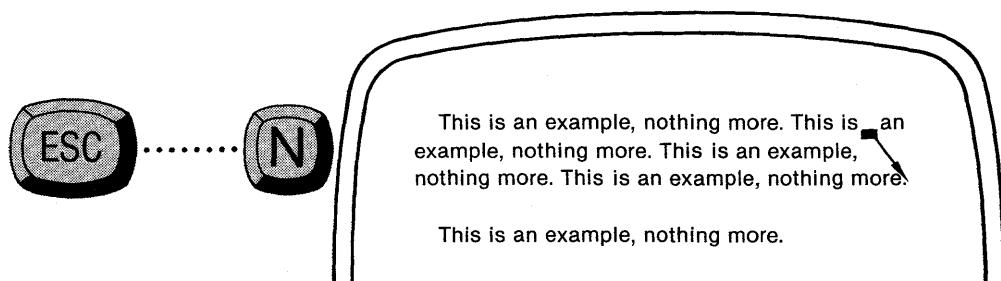
## The BEGINNING OF PARAGRAPH Command

Moves the cursor from anywhere inside the paragraph it occupies to the beginning of the paragraph.



## The END OF PARAGRAPH Command

Moves the cursor from anywhere inside the paragraph it occupies to the end of the paragraph.



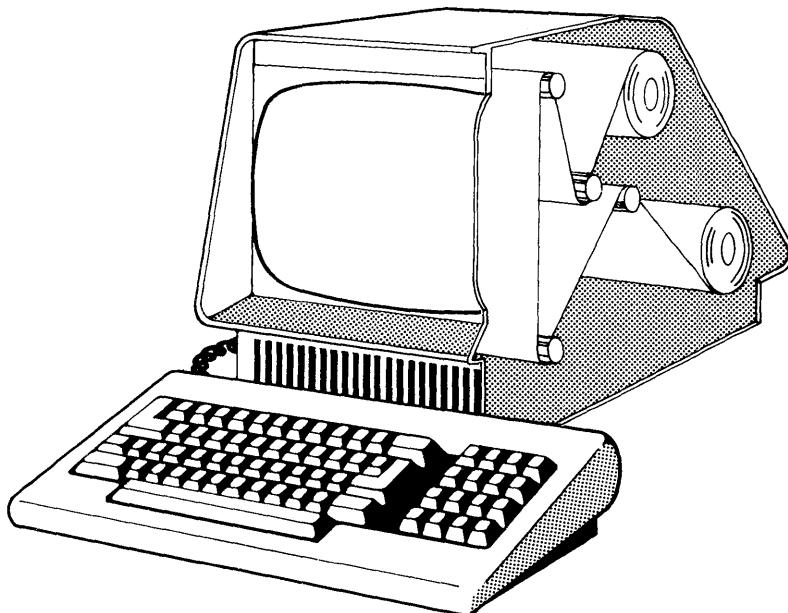
---

Again, note how Control—P and Control—N are related to the Meta commands Escape···P and Escape···N.

## Moving Screens

If your document occupies more than one screen, it is often not convenient to move through it using only the above cursor commands. You will want to move through the text screen-full by screen-full, even jumping to the beginning or end of the document.

To fully understand the commands that will do this, it is helpful to think of the document you are editing as being a continuous roll of film which rises onto the screen at the bottom and leaves the screen at the top, as in the following illustration.



*Figure 3: An imaginative drawing to illustrate the scrolling process.*

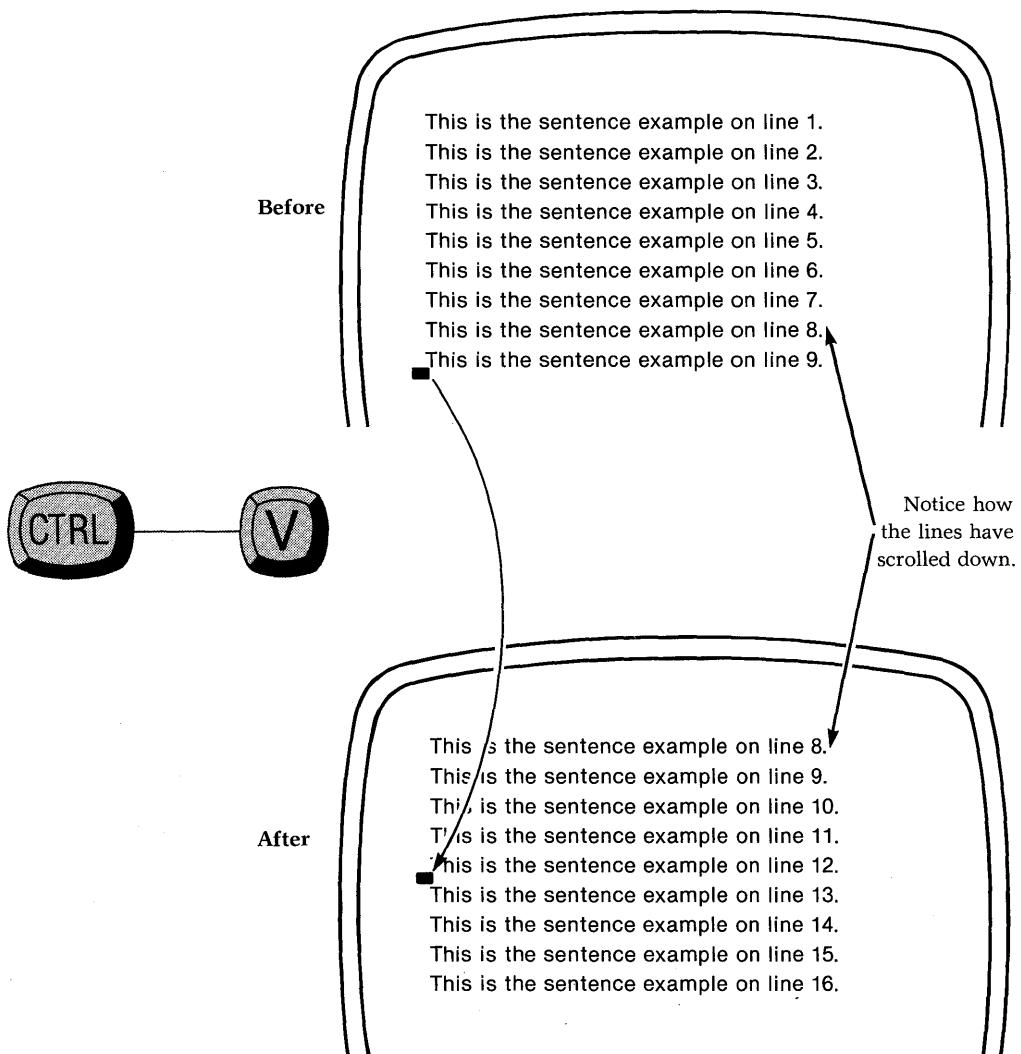
In this illustration, your screen is like a 'window' through which you are able to view portions of the text as it passes before you. The beginning of your document is somewhere 'above' the screen, and the end is somewhere 'below.'

The screen commands which follow allow you to scroll the text in either direction, in order to examine 'previous' or upcoming ('next') pages.

---

## The VIEW NEXT SCREEN Command

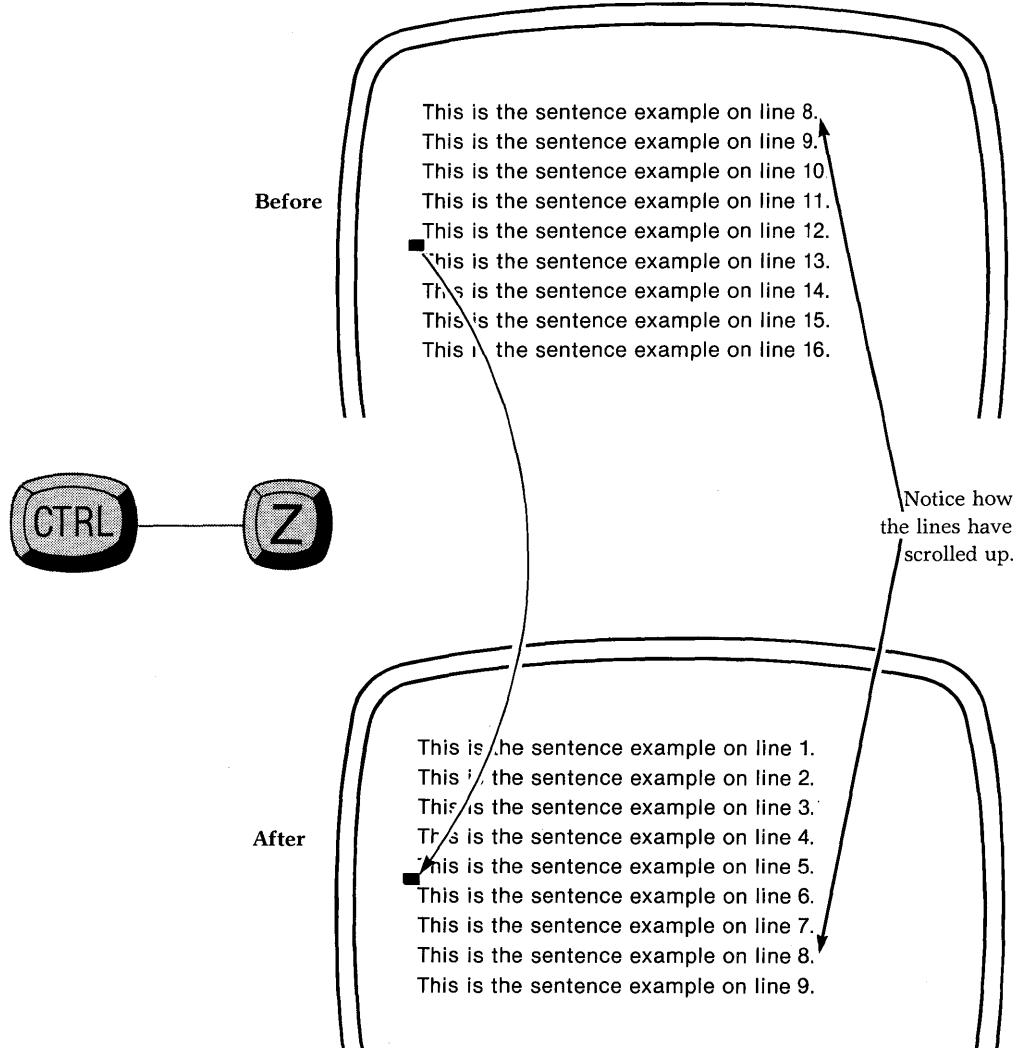
Brings unseen text from the next screen display into view.



## The VIEW PREVIOUS SCREEN Command

Brings unseen text from the previous screen into view.

To provide continuity when going from screen to screen, Perfect Writer will keep the last two lines of the old screen and shift the document up to the next screen.



## The BEGINNING OF DOCUMENT Command

Advances the cursor to the beginning of the document, regardless of the number of lines or length of text.

Before

This is the sentence example on line 39.  
This is the sentence example on line 40.  
This is the sentence example on line 41.  
This is the sentence example on line 42.  
This is the sentence example on line 43.  
This is the sentence example on line 44.  
This is the sentence example on line 45.  
This is the sentence example on line 46.  
This is the sentence example on line 47.



Cursor is  
returned to  
the beginning  
of the  
document

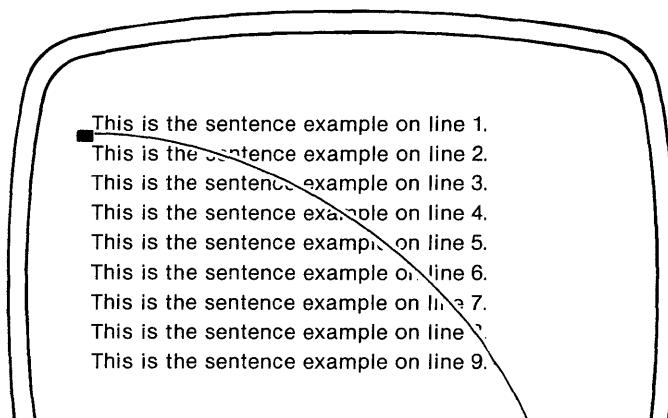
After

This is the sentence example on line 1.  
This is the sentence example on line 2.  
This is the sentence example on line 3.  
This is the sentence example on line 4.  
This is the sentence example on line 5.  
This is the sentence example on line 6.  
This is the sentence example on line 7.  
This is the sentence example on line 8.  
This is the sentence example on line 9.

## The END OF DOCUMENT Command

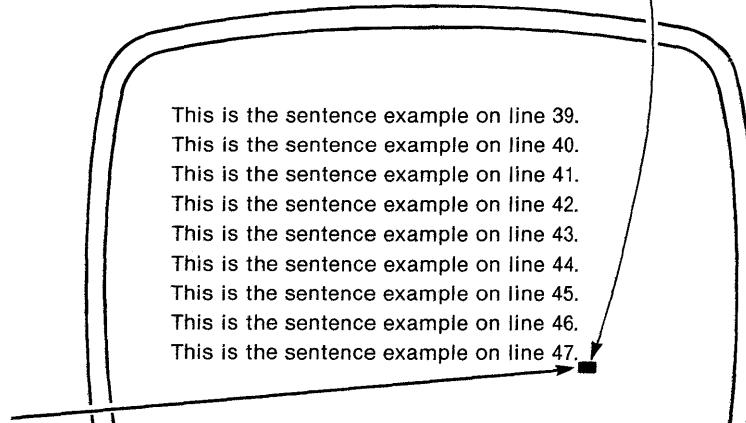
Advances the cursor to the end of the document, regardless of the number of screens or length of text.

Before



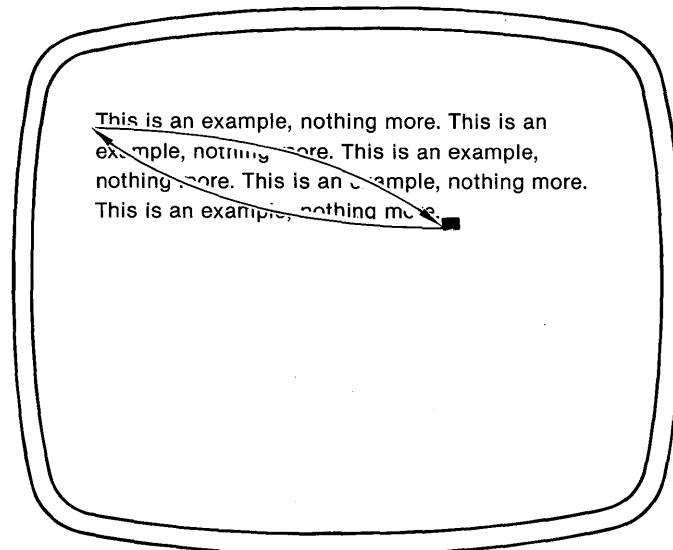
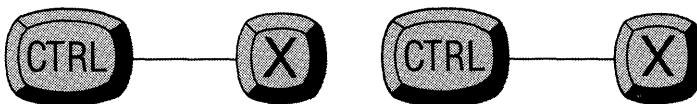
After

Cursor is moved to the  
end of the document



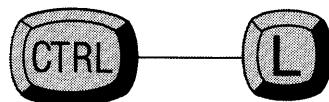
## The EXCHANGE CURSOR & MARK Command

Exchanges the point of the cursor with an invisible mark. Useful for returning to one's original position in a document after executing, for example, a BEGINNING (or END) OF DOCUMENT COMMAND.



## The CENTER TEXT Command

Causes text surrounding the cursor to be redisplayed in the center of the screen. This command is extremely useful with cursor commands. For example, the cursor having been positioned on a line at the bottom of the screen, this command will cause text to be redisplayed so that the line and the cursor are now in the center of the screen.



This is the sentence example on line 1.  
This is the sentence example on line 2.  
This is the sentence example on line 3.  
This is the sentence example on line 4.  
This is the sentence example on line 5.  
This is the sentence example on line 6.  
This is the sentence example on line 7.  
This is the sentence example on line 8.  
This is the sentence example on line 9.

### CURSOR Command Summary:

Forward Character



Forward Word



Beginning of Line



Beginning of Sentence



Beginning of Paragraph



Beginning of Document



▲ Previous Line



Backward Character



Backward Word



End of Line



End of Sentence



End of Paragraph



End of Document



▼ Next Line



### SCREEN Command Summary:

View Previous Screen



Exchange Cursor and Mark



View Next Screen

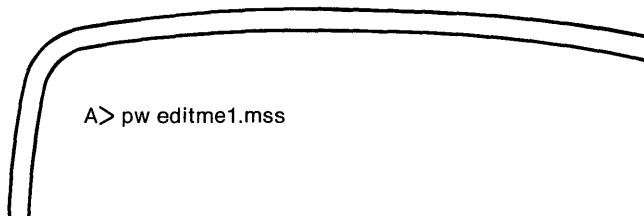


Note: The commands are found on the Reference Card.

---

**Exercise:**

Call up Perfect Writer's exercise text, EDITME1.MSS, by typing:



followed by a 'carriage return'

- On this document practice moving the cursor and centering text, using the various cursor commands described above.

### **EDITME1: Learning the Commands**

Probably the best way to learn the Cursor commands is by remembering that the command character represents in most cases the action that the command is supposed to produce: i.e. "F" for forward, "B" for backward, etc.

Notice also that the Control key commands and the Escape key (or Meta) commands are related, in that in many cases the Meta commands simply produce the same action only on a larger scale. For example, "Control ---- F" moves the cursor forward one character, while "Escape ---- F" moves the cursor forward one 'word.' This relationship holds true for a great many of Perfect Writer's commands, not just those that move the cursor. It is a design feature of Perfect Writer that will help you learn the commands quickly.

As you have probably guessed, you don't have to learn all of the commands right away to begin using Perfect Writer. Learn them as you need them. This takes the pressure out of learning about Perfect Writer. For example, you really only need to know FOUR commands to move the cursor anywhere you want (though it will be slow). Which ones are they?†

---

†They are FORWARD CHARACTER, BACKWARD CHARACTER, PREVIOUS LINE, and NEXT LINE.

It could happen that you will never learn by heart all the commands Perfect Writer offers. But this is all right, because you really don't need to. Some of them you will probably use only rarely, but they are there if you need them. We find that we use nearly all of the commands ourselves, which is why we include them here for you.

Have you removed the Reference Card from your User Manual yet? This card is handy to keep near your keyboard. It provides a very quick reference to all of Perfect Writer's commands.

Practice using the above MOVE SCREEN commands. Jump to the end of the document and the beginning of the document from different locations within the document. Return to your original cursor location using the EXCHANGE CURSOR & MARK COMMAND.

## Chapter V DELETING AND INSERTING

You have found an item to be changed and have moved the cursor to the proper position. Several different delete commands are now available to you. In the following examples note that **the material highlighted in reverse will be deleted** from your screen.

### DELETE Key

This key may be labeled 'DEL,' 'RUBOUT,' or 'BACKSPACE' on your keyboard. It erases the last character typed and moves the cursor back one space.



This is an example, nothing more. ■

Note: For the illustrative purposes of this manual, material highlighted in reverse is deleted.

Deletes previous character

### DELETE PREVIOUS WORD

This command deletes the first word **preceding** the cursor. Position the cursor after the word to be deleted. Type the DELETE PREVIOUS WORD command:



.....



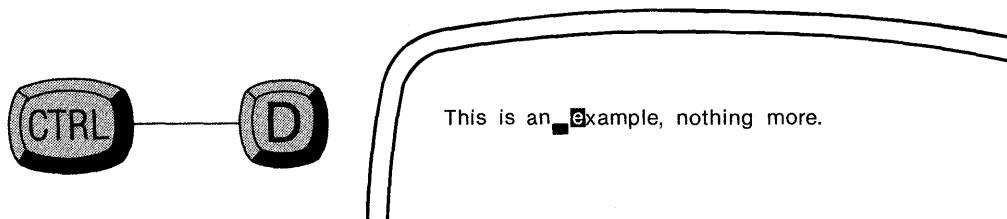
This is an **example**, nothing more. ■

Deletes word preceding cursor

Perfect Writer deletes the previous word and moves the cursor and remaining portion of the line to the left.

## DELETE NEXT CHARACTER

Deletes the next character after the cursor, moving the remaining characters on the line one space to the left. To delete a character, position the cursor before the character to be deleted. Type the DELETE NEXT CHARACTER command:

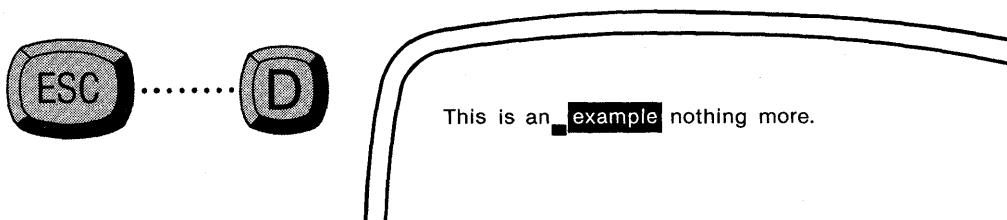


Deletes next character

To continue deleting any number of succeeding characters, continue typing 'D' while depressing the CONTROL key.

## DELETE NEXT WORD

Erases the word immediately following the cursor and moves the remaining portion of the line a corresponding number of spaces to the left. To delete a word, position the cursor before the word to be erased. Type the DELETE NEXT WORD command:



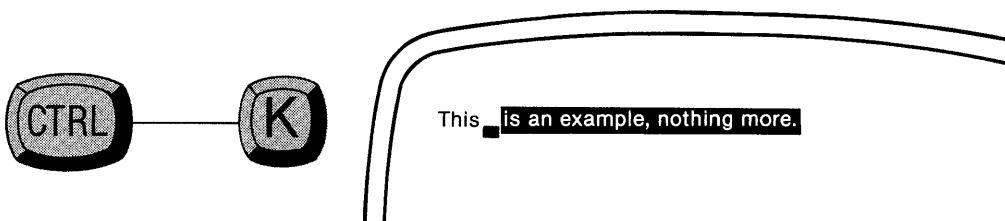
Deletes next word

Note the relationship between Control —— D and Escape ··· D.

---

## DELETE LINE

Deletes all characters from the present position of the cursor to the end of the line. To delete a line, position the cursor at the beginning of the line to be deleted. Type the DELETE LINE command:



Deletes from cursor to end of line

Perfect Writer deletes all characters to the right of the cursor.

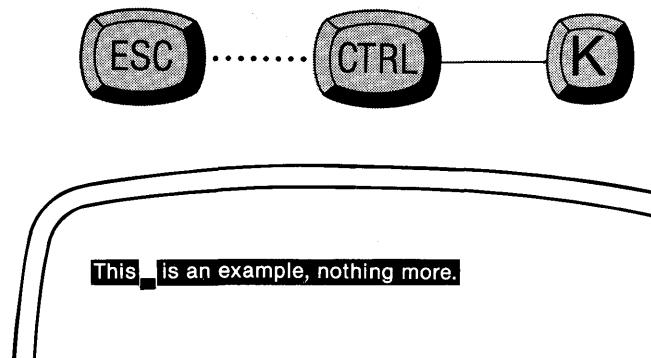
**Note:** Executing the command a second time will erase the blank line itself and move all other lines up one line.

Typing the DELETE LINE command with the cursor placed in the middle of a line will cause only the characters to the right of the cursor to be erased.

## DELETE ENTIRE LINE

Deletes the entire line which the cursor is currently on.

Similar to the DELETE LINE command (Control ----- K), except that this command deletes text on **both** sides of the cursor. To delete an entire line, position the cursor anywhere in the line to be deleted. Type the DELETE ENTIRE LINE command:

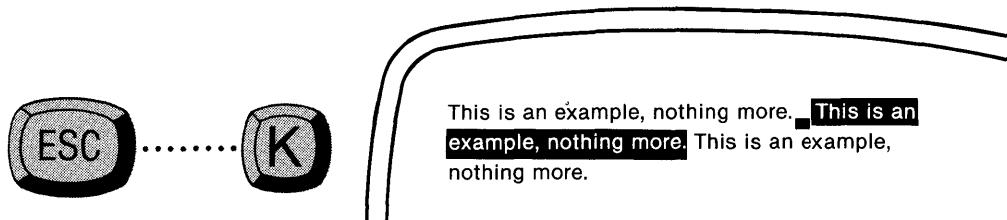


Deletes entire line

Perfect Writer deletes all characters on the line and the line itself.

## **DELETE SENTENCE FORWARD**

Deletes text forward from the position of the cursor to the end of a sentence—i.e., until a period (.), question mark (?), or exclamation mark (!) is reached. To use, position the cursor at the beginning of the sentence to be deleted. Type the DELETE SENTENCE FORWARD command:



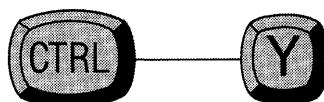
Deletes entire sentence

All text from the position of the cursor to the end of the sentence is deleted, including the punctuation mark.

## YANKBACK—a Safeguard

If you delete anything larger than a character, Perfect Writer saves it temporarily, in case you make a mistake, or change your mind. Your deletion is stored in a 'save buffer' and can be recalled using the YANKBACK command.

The YANKBACK command:



Perfect Writer restores the deleted material to the right of the cursor, moving existing text over to make room. If you have moved the cursor, position it again where you want the deleted text to be restored.

**Note: Normally Perfect Writer saves only the most recent deletion.** That is, if you make a deletion, move the cursor, and then make another deletion in another place, the **first deletion will be lost!** Thus, you must decide to replace a deletion, before continuing to delete other text elsewhere.

This action is reflected in the Mode Line. Upon saving a deletion Perfect Writer displays a plus-sign, '+' in the Mode Line. As long as this '+' is present, Perfect Writer is continuing to save sequential deletions that you make. When the '+' disappears, the saving process has ended. Any further deletions that you make after the '+' has disappeared will cause the previous deletions to be lost! (See Chapter XI, page 8.)

The YANKBACK command is used in copying and moving portions of text either within a document, or between documents. For a discussion of this function, see Chapter XI.

## The WIPE REGION Command

Erases an entire block of text at one time. Before the command can be executed, the region to be erased must be identified by boundary marks, using the MARK BOUNDARIES procedure (steps 1-3 following):

### Steps:

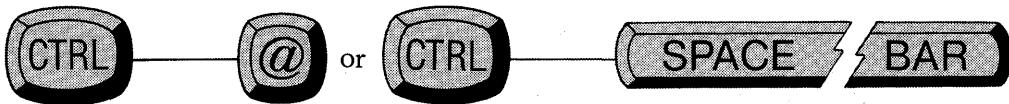
1. Place the cursor on the first character in the region to be erased.
2. Set the forward boundary mark by typing the MARK SET command, which will be **one** of the following (whichever works for your terminal†):



■ This is an example, nothing more. This is an example, nothing more.  
This is an example, nothing more. This is an example, nothing more. This is an example, nothing more.  
This is an example, nothing more.

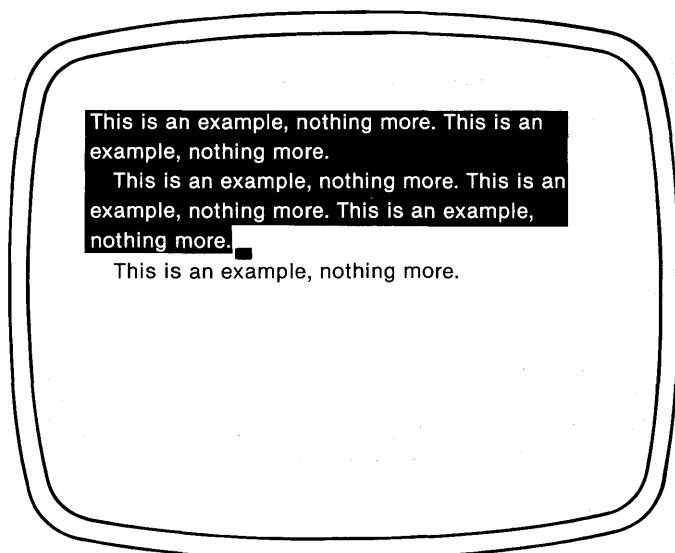
Mark Set

†Also try:



One of these commands will cause Perfect Writer to respond with the message "Mark Set" in the Echo Line, indicating that a forward invisible boundary mark has been set at the beginning of the region to be erased. (This message will disappear when you move the cursor to set the other boundary.)

3. Move the cursor to the end of the region to be erased, placing it just after the last character. The cursor constitutes the rear boundary marker.
4. Type the WIPE REGION command:

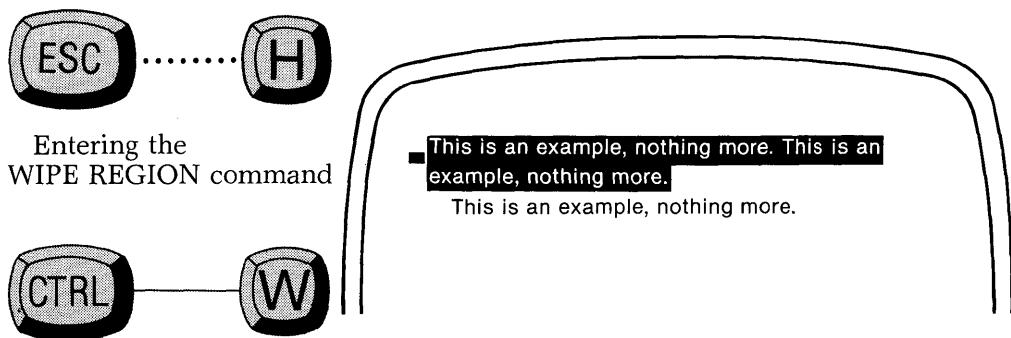


Perfect Writer removes the region of text between the invisible mark and the cursor.

---

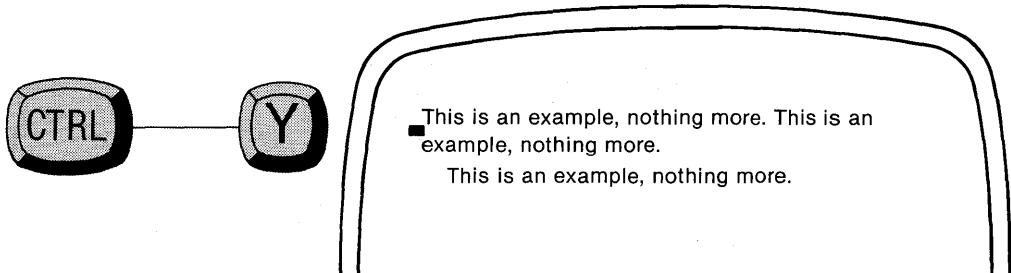
## MARK WHOLE PARAGRAPH Command

Defines or marks the paragraph which the cursor is in. It places the mark at the end and the cursor at the beginning of the paragraph. It is used with the WIPE REGION and YANKBACK command for copying or moving paragraphs. Position the cursor anywhere within a paragraph. Type the MARK WHOLE PARAGRAPH command:



will remove the paragraph.

Entering the YANKBACK command



Note: the deleted text is now restored

will restore the paragraph.

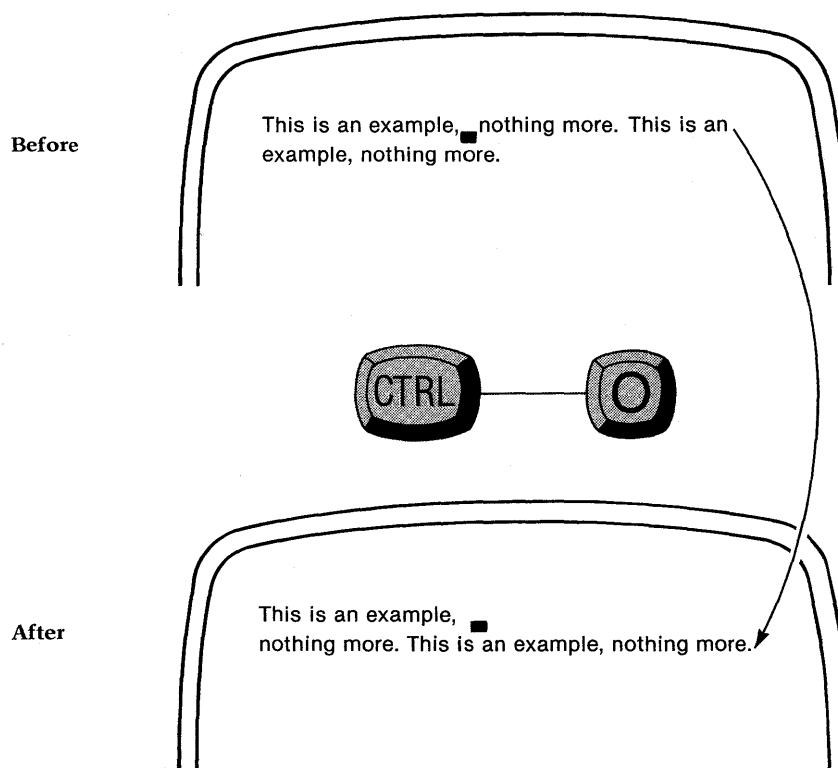
If the cursor is moved somewhere else and the YANKBACK command given, the deleted paragraph will be inserted at the cursor position. It is also possible to switch to a different buffer and copy the deleted paragraph into the different buffer.

## INSERTING

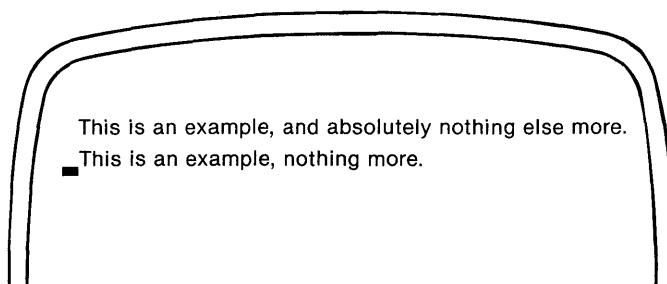
Perfect Writer automatically inserts new words and characters into your text as you type them, moving all original text to the right.

The following steps should be routinely followed when inserting new text:

1. Move the cursor to the point in the text where the new material is to be inserted.
2. Begin typing the insertion.
3. If you find it annoying to watch adjacent text being shifted as you insert, execute the OPEN INSERT command, which will create a new blank line for your insertion. With cursor at the point of insertion, type the OPEN INSERT command:



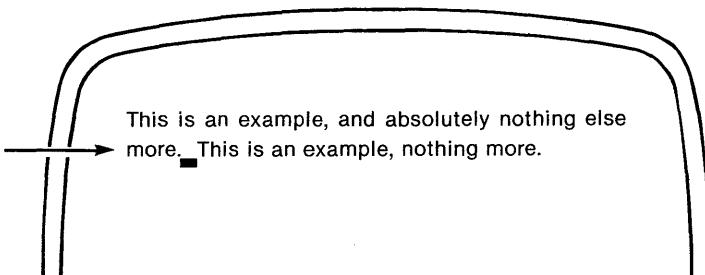
Perfect Writer creates a new blank line for your insertion, by placing an invisible 'newline' character just ahead of the cursor. To close lines again following the insertion, type the CLOSE INSERT command:



4. Following your insertion, adjust the paragraph to which text has been added, by typing the ADJUST TEXT command. Place the cursor anywhere inside the paragraph to be adjusted.



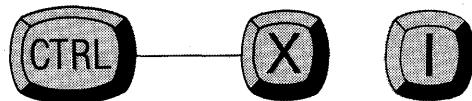
Notice how  
'more' has  
been placed  
on the next  
line to have a  
smooth right  
edge on the  
text.



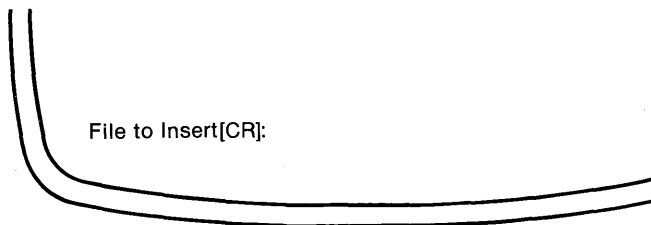
Perfect Writer automatically 'fills' short lines and 'wraps' longer lines in order to make the text uniform.

## The INSERT FILE Command

Perfect Writer allows you to insert a file anywhere in your document. Simply position the cursor where you want the file inserted and enter:



Perfect Writer will respond in the Echo Line:



Enter the filename and press the carriage return. Perfect Writer will insert the file into your current file buffer at the position of the cursor. The cursor is left at the end of the inserted file. The inserted file is also placed in the temporary storage buffer and can be 'yanked back' elsewhere.

## **DELETE AND INSERT Command Summary**

**Delete Previous Character**



**Delete Next Character**



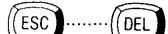
**Delete to End of Line**



**Delete Sentence Forward**



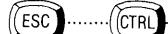
**Delete Previous Word**



**Delete Next Word**



**Delete Entire Line**



**Mark Set**



**Wipe Region**



**Yankback**



**Open Insert**



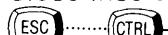
**Insert File**



**Mark Whole Paragraph**



**Close Insert**

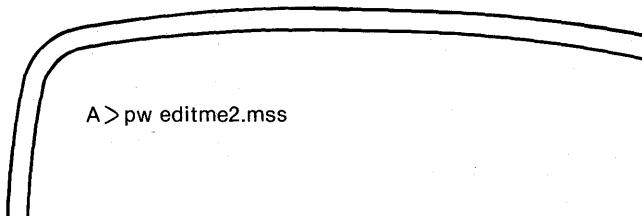


Note: The commands are found on the Reference Card.

---

**Exercise:**

Call up Perfect Writer's exercise text, "EDITME2.MSS" by typing:



followed by a carriage return

On this text practice deleting and restoring words and sentences:

1. Delete the first six characters of the first line. Can you 'yank' them back? (No, because Perfect Writer does not save individual characters.)
2. Delete the first two words, using the DELETE NEXT WORD or DELETE PREVIOUS WORD commands. Can you yank these words back? (Yes! Perfect Writer temporarily saves deletions that are larger than a single character.)
3. Delete a line or portion of a line giving the DELETE LINE command. (Notice that if the cursor is in the middle of a line only the characters to the right of the cursor are deleted.)
4. Delete an **entire** line, after first positioning the cursor at the **end** of the line.
5. Delete a sentence.
6. Delete the first seven lines, afterwards restoring them using the YANKBACK command.
7. Delete the entire document! Which of the MARK SET commands work for your terminal?

### **EDITME2: Practicing Deletions**

Mark Twain gave probably the best advice ever on writing and editing when he said "When in doubt, strike it out!"

Of course, Mark Twain did not have Perfect Writer then to help him delete things. He certainly would have been amazed at how quickly and neatly deletions from a text can be made now. No more having to work around lines that have been 'eX'd out.' No more having to write back IN something you have mistakenly scratched over. (Perfect Writer temporarily saves all deletions larger than a single character.)

Note that the Meta (or Escape) and Control commands share for the most part the same relationship here as they did for moving the cursor. That is, while a Control ----- D will delete the next character, Escape ----- D will delete the next word.

Again, you don't have to learn all the commands at once. A few to start will serve you well for a long time.

When you are finished enter the QUIT command:



## Notes

## Chapter VI **STORING YOUR TEXT**

### **Introduction**

You have finished creating and editing a document, and now wish to have Perfect Writer save it for later editing and printing. Perfect Writer will save your document as a 'file,' identifying it with a unique 'filename.' In saving your document as a file, Perfect Writer creates in effect a 'master copy' of the document, which cannot thereafter be erased or destroyed unless you specifically request it.

When you wish to edit a document you have previously saved, Perfect Writer places a **copy** of that document into a temporary working space for you. **Whatever changes you make to this copy do not affect your original.**

When you have finished editing the copy, you have the option of either: replacing the original with the newly edited copy, or of saving the copy in a new file by itself, thereby retaining **both** versions of your document.

Perfect Writer allows you to create as many new files as storage space will allow. To help you remember the files you have created, the DOS operating system will allow you to list the names of all files on your disks.

## File Names

DOS requires that file names follow a particular form. Specifically, file names cannot be longer than eight (8) characters, and cannot contain any of the following > < . , ; = ? \* \_\_. It is helpful when creating file names to be as inventive as possible. Some examples of file names are:

CHAPTER3.MSS (Chapter three of your book)

MAYDIARY.MSS (May entries of your diary)

JSSMITH.MSS (The "J S Smith" file)

## Default File Name

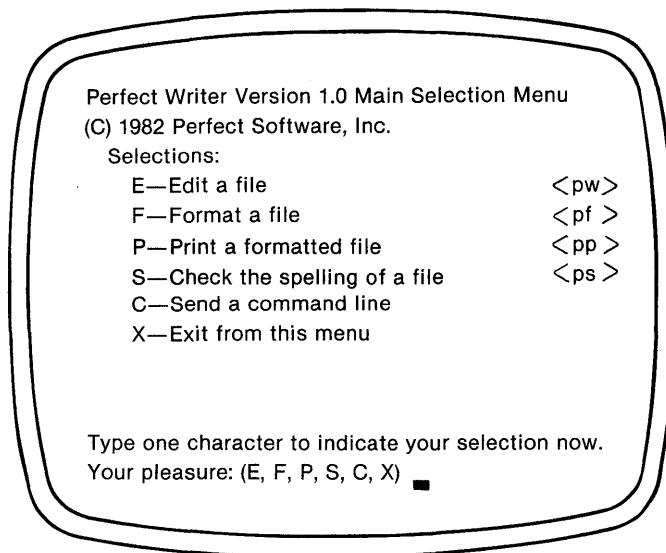
Should you neglect to supply a filename, Perfect Writer will supply a 'default' filename, one that is easily recognized: "NAME.ME." This filename indicates that the document contained in it has not been given a name.

**Important:** The 'NAME.ME' file is a temporary 'back-up' file into which Perfect Writer will continue to store unnamed documents. However, each time it stores a document, **it overwrites whatever may have been stored there previously.** Don't risk losing your documents by not giving them file names!

As a reminder of which document you are working with, Perfect Writer displays the current file name in the Mode Line. If you have not yet named the document, the default filename, 'NAME.ME,' will be displayed.

## **Creating A New File**

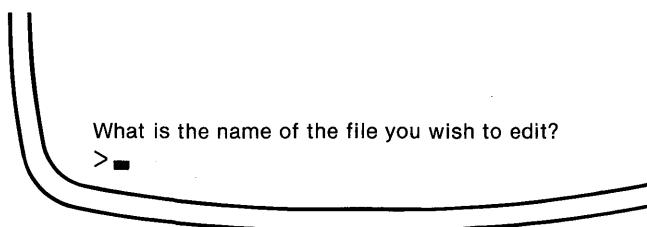
You can create a new file at the time you enter the Perfect Writer system, simply by selecting to 'edit a file' from the main selection menu. For example, from the main menu:



Enter: e

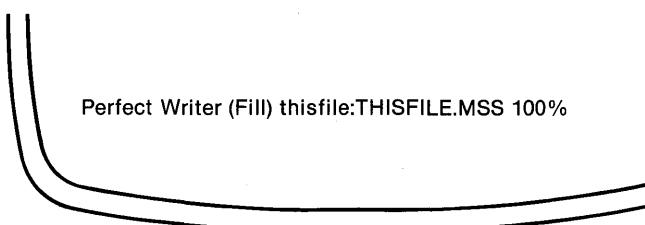
---

Perfect Writer will respond



Enter: THISFILE.MSS [CR]\*

"THISFILE.MSS" is the name of a new file you wish to create. When Perfect Writer comes on-line, it will present you with a blank screen, ready for whatever document you wish to enter into "THISFILE.MSS". The Mode Line will appear as follows:



---

\* You may enter more than one file into editing buffers when entering the Perfect Writer editor. When you list several files to be placed in buffers, then Perfect Writer places the last listed file on the screen.

## **FILE COMMANDS**

The following commands instruct Perfect Writer to retrieve or save files:

### **The READ FILE Command**

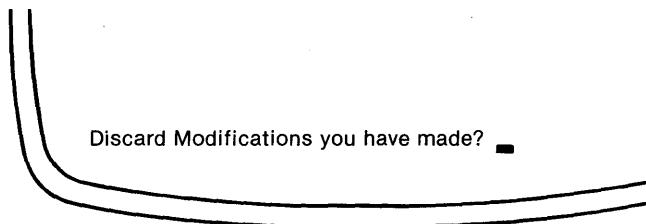
Retrieves a stored file document and places it in the buffer currently being used, overwriting whatever document is presently there. (Useful for 'starting over' in the editing of a document.)

#### **Steps:**

1. At any time during editing type the READ FILE command:

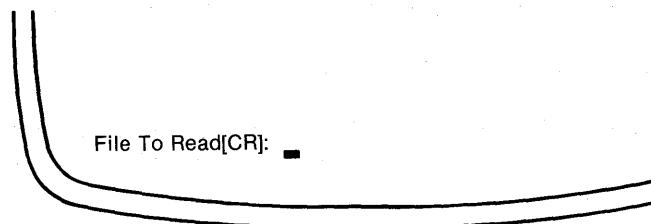


If the document currently being edited has not yet been saved (via the SAVE FILE command or the WRITE FILE command), Perfect Writer responds with the message:



2. Typing "NO" cancels the command and returns you to your place in the text (in essence giving you the opportunity to save the current document.)

3. Typing "YES" causes Perfect Writer to respond with the message:



4. Type the name of the file you wish Perfect Writer to retrieve. Perfect Writer will read the named file into the current buffer, overwriting whatever document is there.

## The SAVE FILE Command

Immediately copies the document being edited to the disk file identified in the Mode Line. (This command, if used frequently, guards against accidental losses of the text being edited.) At any time during the editing process, type the SAVE FILE command:



Perfect Writer immediately copies your document to the disk file named in the Mode Line, **overwriting your original** or whatever version of the document was there before. The system message 'Writing' is displayed briefly, followed by 'File Written.'

After safely storing your document Perfect Writer returns you to the text to continue editing.

**Note:** Using the SAVE FILE command causes the original 'master' copy of the document, if one existed, to be overwritten. Also, if you should get a "BDOS ERROR ON DISK" message, which means that Perfect Writer has encountered some problem with the disk (perhaps it is faulty), hit the "Enter" key, or the GO BACK command (Control ---- G). This will hopefully restore you to your file in the Editing Mode of Perfect Writer. When the "File Written" message appears, either copy the file using a different filename (see the next command), or copy it onto a different disk.

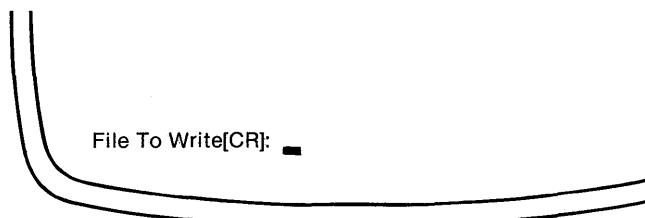
---

## The WRITE FILE Command

Saves the document currently being edited. However, this command gives you the option of either overwriting the old 'master' copy, or of storing your updated document in a **different file** under a **new name**. At any time during editing, type the WRITE FILE command:



Perfect Writer responds with the following message in the Echo Line:



You have two options:

- Option 1:** You can create a new and different file, and Perfect Writer will store the document in that file. [Type the new filename followed by a 'carriage return.']}
- Option 2:** You can instruct Perfect Writer to overwrite the old 'master' copy. [Simply type a 'carriage return.')] Perfect Writer immediately copies your document into the file identified in the Mode Line, overwriting what was previously stored there. (This second option is identical to the SAVE FILE command.)

## **Retrieving Your File**

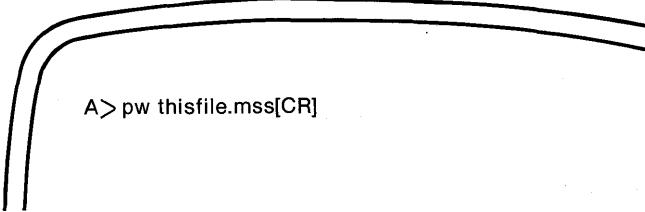
There are several methods of retrieving documents that have been stored as files. However, in this Basic section we will present only one, which must be used at the time you enter the Perfect Writer system.

### **Basic File Retrieval**

You are about to enter Perfect Writer and wish to edit a file that you have previously saved.

#### **Steps:**

1. Consult your system directory for the correct name of the file you wish to retrieve. [A > dir]
2. Type the filename after the command calling up Perfect Writer:



A> pw thisfile.mss[CR]

Perfect Writer will retrieve the file and present it to you on the screen. As we discussed earlier, you could also retrieve THISFILE.MSS through the menu program (option e).

## Chapter VII **PRINTING A DOCUMENT**

### **Introduction**

You have finished writing and editing a document and have saved it under a specific file name. You now wish to print it.

Perfect Writer offers two print options: A 'Full Feature' print option, and a 'Quick Print' option. The former allows use of all the typeface commands and the microspacing features your printer is capable of. The Quick Print option allows you to print your document exactly as you have created it on the console screen. In the beginning as you are learning about Perfect Writer, you will probably use the Quick Print option most often, finding it both satisfactory and convenient for a good many of your printing needs. However, when your printing requires any of the special typeface formats, microspacing or proportional spacing, you should use the Full Feature print option.

In their procedures, both print options are closely related, except that the Quick Print option 'shortcuts' a few steps. In this discussion we will first present the Full Feature print option to insure that you understand the basic procedures involved in printing a document, followed by the steps necessary to produce a 'quick print.'

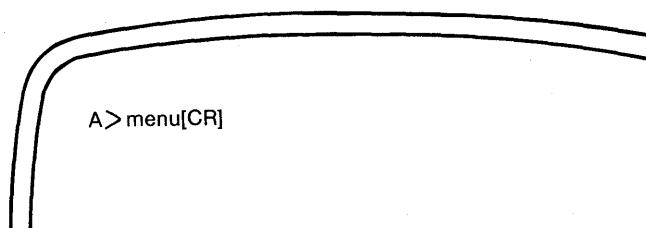
In printing, Perfect Writer invokes two internal programs called "Perfect Formatter" and "Perfect Printer." Working together these two programs prepare and output the final version of your document. The menu options presented in this section represent the selections available to you when using these two programs.

## THE FULL FEATURE PRINT OPTION

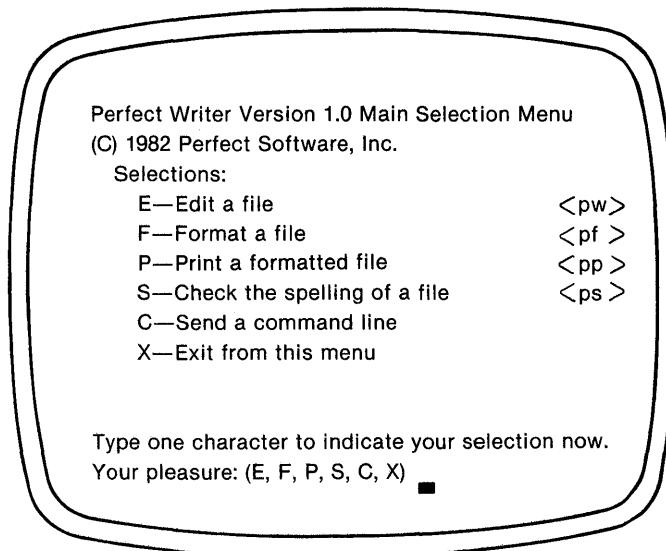
The Full Feature print option allows you to use all of the formatting and printing features provided by Perfect Writer, including:

- use of either a continuous form feed printer or a single sheet feed printer,
- the printing of multiple copies,
- the ability to start printing at any page of your document,
- use of all of the type face format commands.

STEP 1. Call up Perfect Writer's Main Selection Menu by entering:

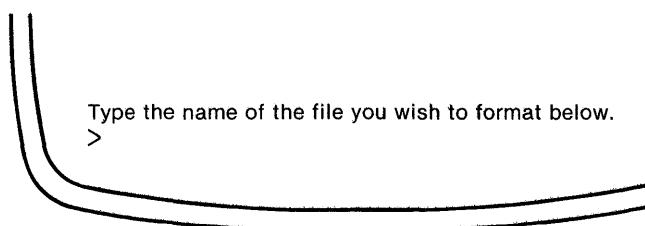


Perfect Writer will present the Main Selection Menu:

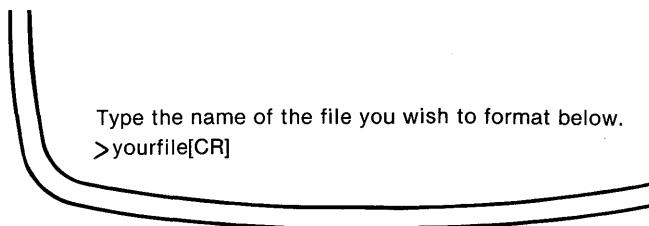


STEP 2. Before any document can be printed, it must be formatted. For this example we are assuming that our file would not previously have been formatted. Therefore our selection here would be "F—Format a File." Simply type the letter "F".

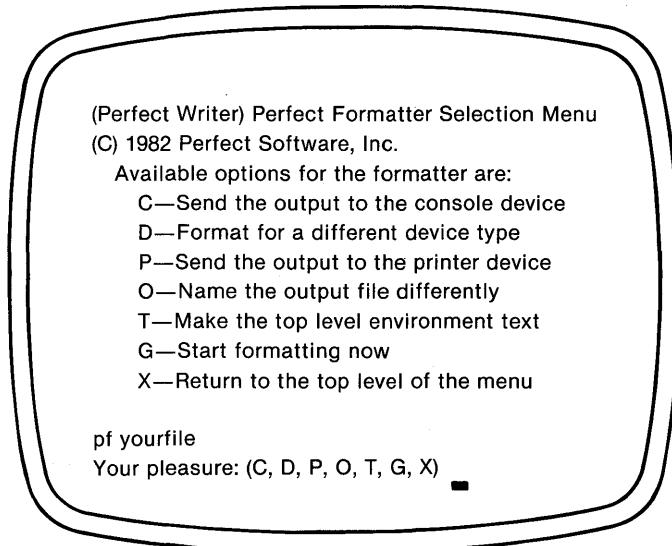
Perfect Writer will respond:



STEP 3. Enter the document's filename. You need not include the extension ".mss" since Perfect Writer will assume this. Also, if the file is on a different disk drive than the program disk containing Perfect Writer (which is the usual situation), then do not forget to specify the drive the document is on. For example:



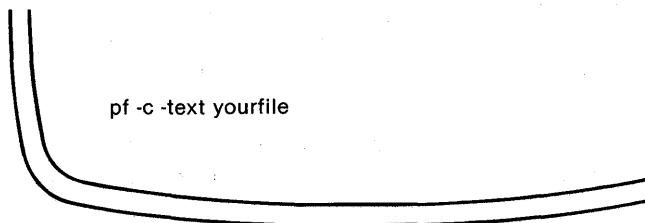
Perfect Writer transfers you to the Perfect Formatter Selection Menu:



When formatting your document Perfect Formatter will create a new intermediate file called a "FIN-file," standing for "FINished" or "FINal." (This file can be identified on your directory by the ".FIN" extension that has been attached to the filename.) The "FIN-file" is used later by Perfect Printer in printing your document. This selection menu presents the several options available in the creation and subsequent use of this "FIN-file." Several require further explanation:

- C — Sent output to the console (That is, while creating the formatted "FIN-file," display it concurrently on the console. With this option no FIN-file will be produced.)
- D — Format for a different device (Normally the FIN-file will be created to meet specifications of the default printing device you have specified during configuration. However, this option allows you to format the file for any alternate device you have defined to Perfect Writer. [See Chapter XVIII, page 1])
- P — Send output to the Printer (This is the Quick Print option. When this option is selected then no FIN-file is produced. Like the "C" option, this option prints the file while formatting it.)
- O — Name the output file differently (This option allows you to give the outputted FIN-file a different name. For example, when formatted "yourfile.mss" might become "thatfile.fin".)
- T — Make TEXT the top level environment. (This option allows you to change the format environment from VERBATIM to TEXT [see Chapter XIV, page 4].)

It is possible to select more than one of these options. Each one that you select will be "echoed" in the menu Echo Line. For example:



Note here that Perfect Writer echoes the name of your file in the menu Echo Line, along with the 'pf' command (which can be used to invoke Perfect Formatter directly without using the menu system.<sup>1</sup>)

STEP 4. For this example, let us assume that we wish only to format the file without selecting any special options. We would select "G—Start formatting now."

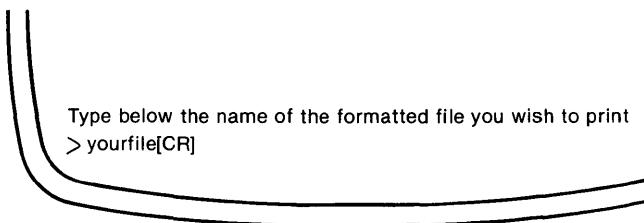
Perfect Formatter automatically creates a "FIN-file," storing it on disk.

---

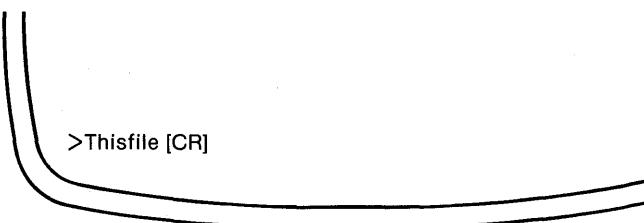
<sup>1</sup>The Perfect Writer menu program has been designed to make Perfect Writer easier for you to use. However, it is not necessary to use the menu program and after acquiring experience with Perfect Writer you may find it an unnecessary aid. To assist you in learning how to use Perfect Writer without the menu program, all commands issued by the menu program are echoed on the screen. What you accomplish using the menu program could also have been achieved by directly entering the commands echoed in the menu. If you enter the commands directly you would save the time it takes to call into your computer's memory the menu program and the time it takes to switch out. You may elect to delete the menu program at any time without affecting the performance of Perfect Writer.

STEP 5. Following formatting you will be transferred back to the Main Selection Menu of Perfect Writer. You are now ready to print your document. Select "P—Print a Formatted File."

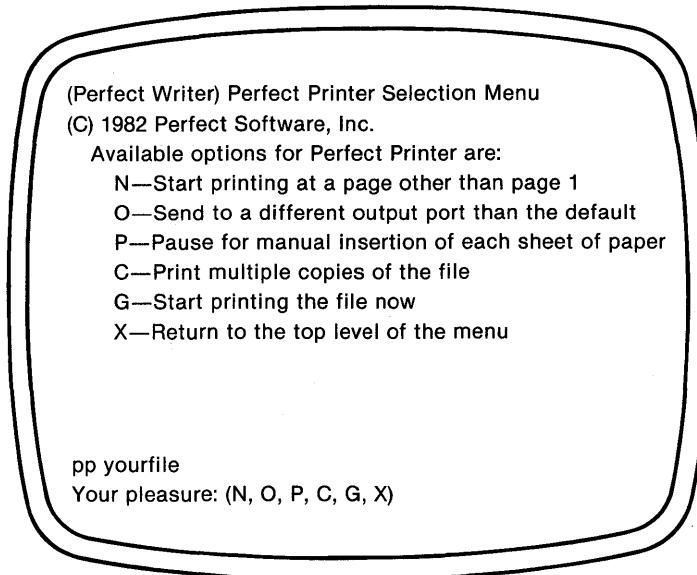
Perfect Writer responds in the menu Echo Line:



STEP 6. Enter the name of your file WITHOUT the extension. (Remember that Perfect Formatter has just created a new, intermediate, formatted file and labeled it "YOURFILE.FIN". When printing this file Perfect Printer will assume the ".FIN" extension. If your file has not been formatted and assigned the ".FIN" extension, then it will not be printed.) Enter:



Perfect Writer transfers you to the Perfect Printer Selection Menu:



This menu presents the various options that are available to you during **printing**. You can select more than one option, but not all. Selecting too many options will result in a "warning" from Perfect Writer saying it cannot do all that you ask. The options are:

**Where to Begin Printing:** It is possible to begin printing in the middle of a document. This option allows you to specify the page on which printing is to begin.

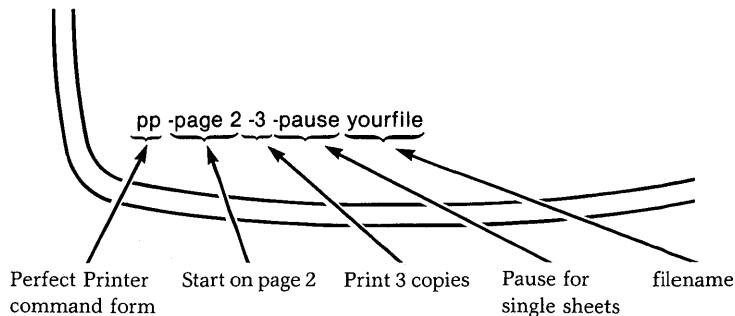
**Use Different Output Port:** This would be used if you had a serial as well as a parallel printer connected.

**Single Sheet Printing:** If you select this option Perfect Printer will "pause" between pages and allow you to insert a fresh sheet of paper before continuing.

**Multiple Copies:** If you wish to print more than one copy of the document, you will be asked to specify the number of copies.

---

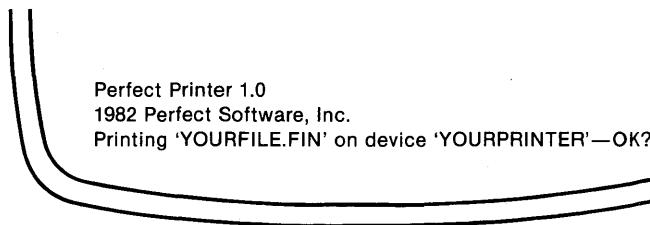
Your selections will be "echoed" in the menu Echo Line, along with your file-name and the command that can be used to invoke Perfect Printer. For example:



STEP 7. When you have selected the printing options, select the option that will commence printing of the document:

"G— Start printing the file now"

Perfect Printer will clear the screen and respond with the message:



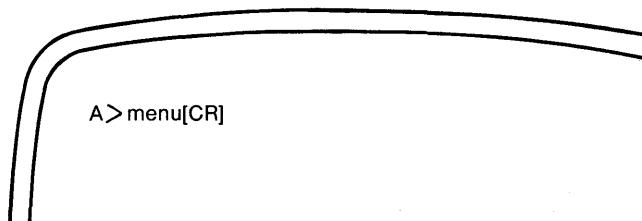
Enter "Y".

As it prints your document, Perfect Printer will echo the page number of the page it is printing on the screen of your terminal. After your document has been printed, Perfect Printer will return you to the Main Selection Menu of Perfect Writer.

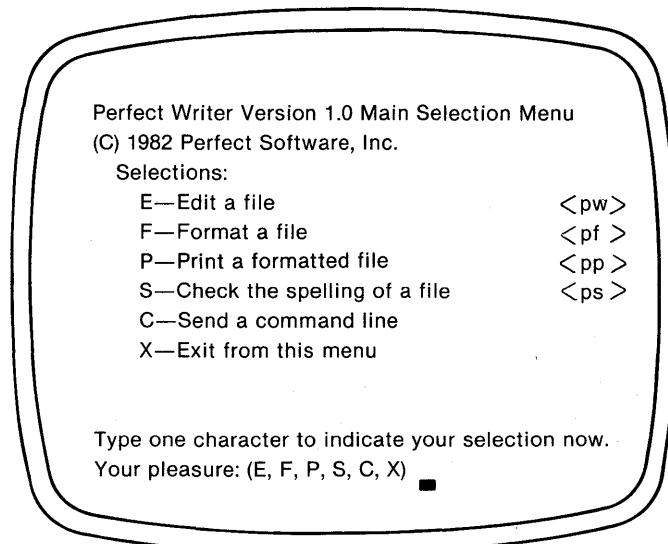
## THE QUICK PRINT OPTION

The Quick Print option is similar to Full Feature Print Option, except that it does not allow printing to begin in the middle of a document nor does it allow the printing of multiple copies. Further, it will NOT permit the use of typeface commands. The first three steps of the Quick Print option are identical with the Full Feature print option.

STEP 1. Call up the Perfect Writer menu system by typing:

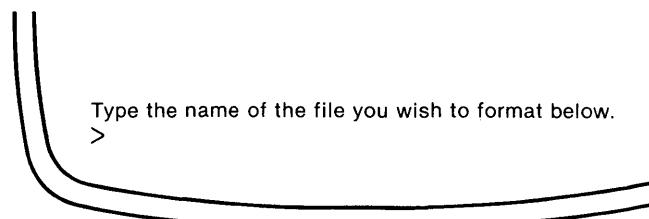


Perfect Writer will respond with the Main Selection Menu:

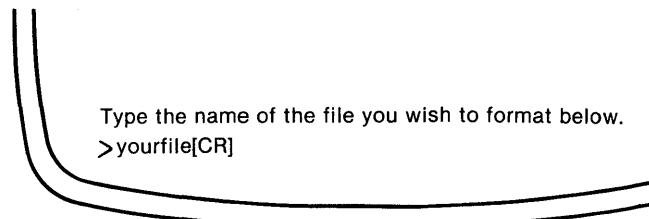


STEP 2. Select "F—Format a File".

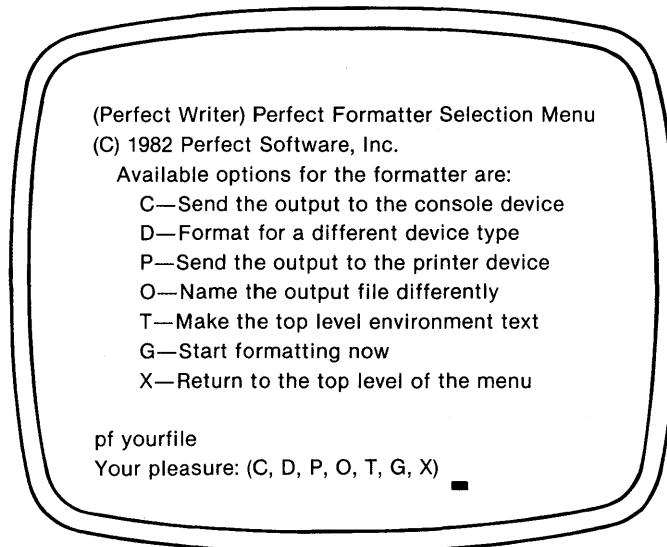
Perfect Writer will respond:



STEP 3. Enter the name of your file, with or without the extension ".MSS"



Perfect Writer will transfer you to the Perfect Formatter Selection Menu:



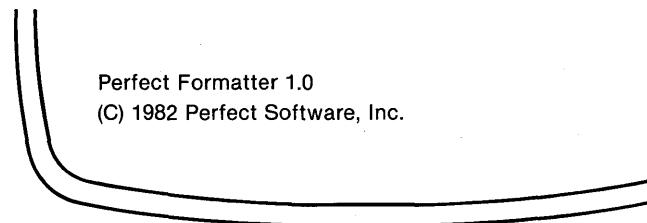
STEP 4. Until now the steps for the Quick Print option have been essentially the same as the Full Feature print option. At this point however, the Quick Print option requires that you make only two selections, in this order:

- a) Select "P—Send the output to the printer". This selection will be echoed in the menu Echo Line:



- b) Select "G—Start formatting now".

Perfect Writer will clear your screen and display the message:



The printer will begin printing your document. As it is printed Perfect Writer will echo the current page it is printing on your terminal. When your document is finished printing Perfect Writer will return you to the Main Selection Menu.

Selecting the Quick Print option instructs Perfect Formatter to print your document WHILE it is being formatted. That is to say, the "FIN-file" is being outputted directly to your printer. Perfect Writer's internal program "Perfect Printer" which allows printing of multiple documents and use of the typeface commands is being bypassed altogether.

---

## STOP PRINTING

Typing the GO BACK (or CANCEL) command at the console at any time during the printing operation will cause Perfect Writer to stop printing the current file document.



## EXERCISE

Print Perfect Writer's practice document "LETTER1.MSS", shown below. Send this letter to a friend by editing it appropriately. Save it, using the SAVE FILE command (Control ----- X Control ----- S). Finally, print your changes using the printing procedures just outlined.

---

(enter your return address,  
Anytown, Anystate, ZIP)

Dear [Your friend's name]:

Surprise! Here's a letter to you in the middle of the day. Actually it's just a practice letter. What I'm doing is testing our new word processor, Perfect Writer, which we've just installed. Of course, you know we've had word processors here before. But let me tell you Perfect Writer is different! (Wow! What a difference! Where has this thing been all my life?!)

Let me tell you about it:

It includes all the standard writing and editing features you might expect on a word processor: a variety of commands that quickly and easily move the cursor; simple deleting and inserting procedures; safe and convenient routines to store the material you've just created.

---

But besides this it offers several really wonderful features I haven't seen before, like: being able to divide the screen into two separate parts, each of which can hold a different document. Using this 'split-screen' you can edit one letter based upon information in another. You can even transfer sections of text between the two. Or you can simply use the two screens to compare two different versions of the same text to see which is better. Can you imagine all that?!

But there's more . . .

Perfect Writer uses something called 'virtual memory,' which means that you don't have to worry any more how long the document is that you want to write and edit. With Perfect Writer I can edit anything from a memorandum to a book! As well, cursor commands allow me to view instantly any part of the document I wish, no matter how long it is. I think that is utterly fantastic.

As far as printing goes, Perfect Writer offers two print options. The first is just your basic printer, which will quickly reproduce on paper whatever you've just created on the screen.

However, when you need to get fancy and 'perfect,' especially with long documents (Oh, don't we love to do those over and over again!), Perfect Writer offers a system that literally does the formatting for you. No more having to set tabs, margins, indentations, or to keep track of paging, footnotes, index entries—all that stuff!

All you have to do is to tell Perfect Writer, with a brief, easily understood command, how you want to format any particular portion of text: as a list, a chapter heading, a quotation, an address, an example—there are more than 50 formatting options provided.

Perfect Writer automatically does what needs doing—centers it, boldfaces it, underlines it, indents it, italicizes it, numbers it, single or double spaces it—whatever! Any changes you might make later are automatically incorporated—say, if you add another footnote or an extra chapter section. These changes are automatically reflected in the Table of Contents and alphabetized index which Perfect Writer automatically produces. Can you believe all this?

I know that this description must make you think that Perfect Writer is something out of the twenty-first century . . . !

But to top it off, Perfect Writer is EASY TO LEARN!

---

The people we ordered it from claim that it is the most advanced word processing system available, written in the most advanced computer language and modeled after the world's most powerful word processor that has until now only been available on large computers. This scared me at first because I imagined I'd have to go through some awful process of learning a complicated computer language—you know, something like 'Fortran' or something.

But this wasn't the case with Perfect Writer. Even though it has a large and flexible set of commands, they're not difficult to learn. In fact, for what they accomplish, they're a lot more understandable and sensible than all the other systems I've seen.

Too, the manual that accompanies Perfect Writer is READABLE! I can actually understand what it's talking about, and it's easy to find things if I have a question. (You know how incomprehensible some of this stuff is they try to sell you!)

(Perfect Writer must be easy to learn if in just one day I've learned to do all this!)

Whew! This is some letter. You can see how enthusiastic I am. I guess for the first time in a long time I feel that I'm learning some new and valuable skill—because these computers are here to stay. (As far as I'm concerned they didn't get here a moment too soon!)

I could go on, but listen, why don't you ride the elevator down during lunch and I'll **show** you what Perfect Writer will do. Then you can start lobbying for one in your department, too. If this place needs one thing, it's more Perfect Writers working for it.

Yours,

## Notes

## Chapter VIII

# MODES AND ADDITIONAL COMMANDS

### **Modes of Operation**

When you begin editing Perfect Writer provides you with a standard editing mode called "fill." In the Fill Mode lines are 'wrapped' when the text you are entering goes beyond the right margin (also called the "fill column"). That is to say, the words that would otherwise extend beyond the right margin are brought back to start a new line, i.e. "wrapped." This makes your text appear neat and orderly on the screen, and there is no need to continually monitor line lengths as you would with a typewriter.

In addition to the Fill Mode, Perfect Writer offers several other editing 'modes.' These additional modes provide features that you may want to use while editing. If so, you can select the particular mode you want with the mode selection commands.

Except for 'Normal Mode,' all of the modes listed below represent modes of functioning that can be ADDED to the standard Fill Mode, which is to say, that Fill Mode will continue to function even though the new mode has been added.

### **Modes Available**

#### **Fill Mode**

Perfect Writer provides the Fill Mode as the standard editing mode. When you enter Perfect Writer you will begin in the Fill Mode. The major feature of the Fill Mode is that words are wrapped around to the beginning of the next line after you reach the right margin (termed the fill column).

#### **Normal Mode**

Normal Mode turns off the 'word wrap' feature of the Fill Mode and allows you to type a line to the right edge of your screen (column 80). It does not automatically wrap text entered. This is a useful feature when you are preparing tables or other text which will not be in the form of paragraphs.

---

## Save Mode

To prevent the inadvertent loss of text entered, Perfect Writer offers the Save Mode. In the Save Mode text is saved (or written on your disk file) after every 512 characters are typed in. The limitation of this mode is that it slows down text entry because it requires a pause while saving the text. When a document becomes several pages long, the saving operation takes longer and is more inconvenient.

## View Mode

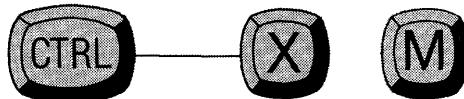
The View Mode allows you to examine text without being able to alter it. If you want to be sure that a document remains unchanged while you review it with the editor, this is the mode you would select.

## Overwrite Mode

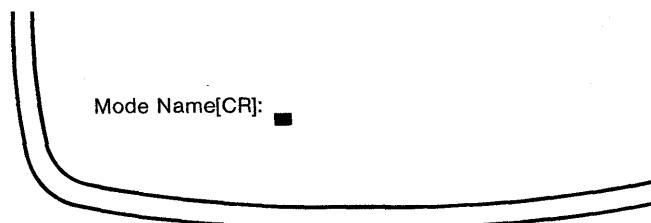
Overwrite Mode turns Perfect Writer into a different kind of word processor, where the cursor always moves vertically and typed characters overwrite instead of inserting.

## The ADD MODE Command

The following command allows you to add the features of the selected mode. To add a mode, enter:



Perfect Writer will respond by asking for the mode name in the Echo Line:



Enter the mode desired and it will be added.

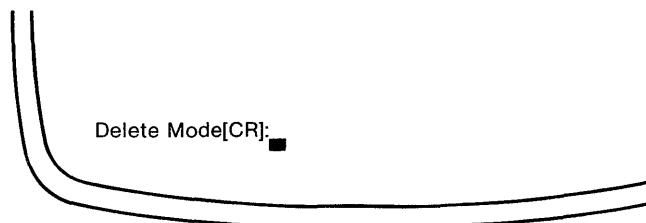
---

## The DELETE MODE Command

To delete a mode simply enter:



Perfect Writer will respond:



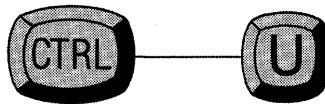
Enter the mode you want to delete and it will be deleted.

## The REPEAT Command

Causes other commands (or characters) to repeat. The number of repetitions may be specified, but if it is not, a default value of four (4) is supplied by Perfect Writer, causing the other command to repeat **four** times.

### Steps:

1. Type the UNIVERSAL REPEAT command:

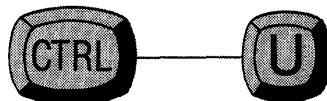


Perfect Writer responds with the message:



2. Enter the number of repetitions desired. The default value of '4' repetitions will be replaced with the number that you type.
3. Enter the command (or character) you wish to have repeated. Perfect Writer immediately repeats the command or character.

**Example 1:** You wish to draw a line across the page. Type the UNIVERSAL REPEAT command:



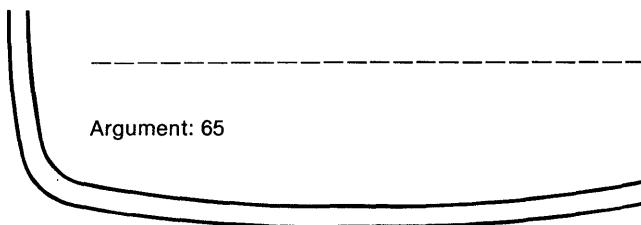
followed by:

'65' (the width of a page)

followed by:

'—' (a single dash)

Perfect Writer will produce a line of dashes across the screen, thus:



## META REPEAT

Typing Escape followed by a number (1,2,3,...99) is an alternative method for repeating a command. Thus entering:

Escape 65 -

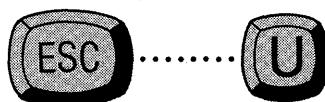
would produce the same result as in example 1 above.

## The UPPERCASE WORD Command

Changes the characters of a word to uppercase.

### Steps:

1. Position the cursor before the word to be 'uppercase.'
2. Type the UPPERCASE WORD command:



Perfect Writer changes each letter in the word to uppercase and leaves the cursor at the end of the word.

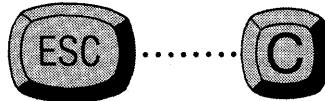
## The LOWERCASE WORD Command

Lowercases all letters in the current word. Follow the same procedure used for UPPERCASE WORD command.



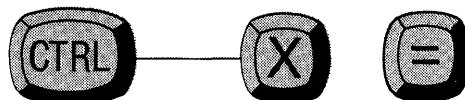
## The CAPITALIZE WORD Command

Capitalizes the current word. Follow the same procedure used for UPPERCASE WORD command.

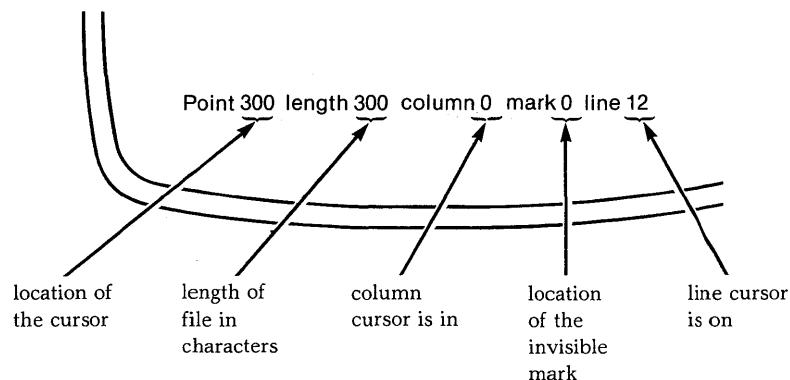


## The LOCATION Command

Displays the location of the cursor point (in characters from the beginning of the buffer), the length of the buffer in characters, the current column, the location of the mark (in characters from the beginning of the buffer), and the current line number. Enter:



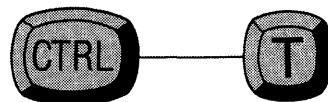
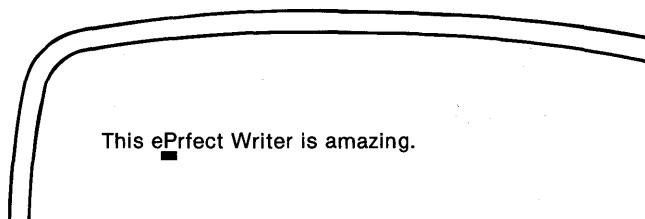
Perfect Writer will respond:



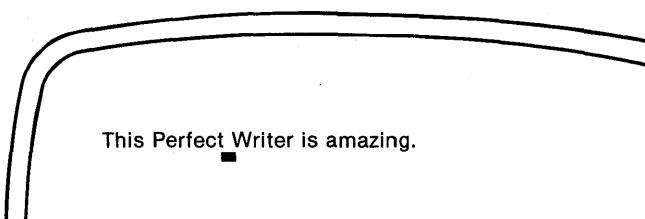
## The TRANSPOSE CHARACTERS Command

Switches the character the cursor is on with the character before it, leaving the cursor after the second. Position the cursor on the second of the two characters to be switched. Type the TRANSPOSE CHARACTERS command:

Before



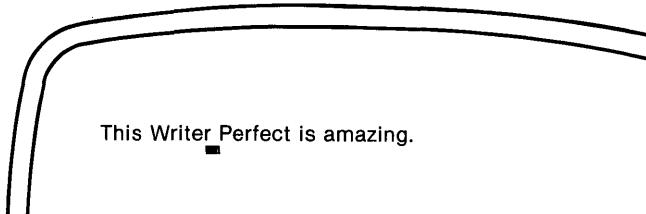
After



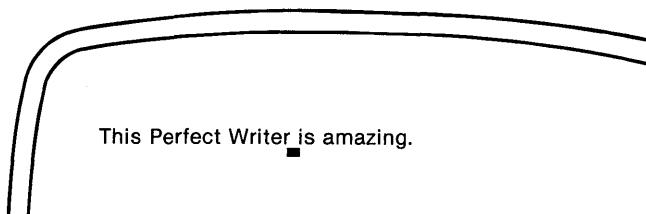
## The TRANSPOSE WORDS Command

- Transposes the two words before and after the cursor, leaving the cursor after the second. Position the cursor between the two words to be transposed. Type the TRANSPOSE WORDS command:

Before



After



**Note:** If the cursor is at the end of a line, the last word on the line is exchanged with the first word on the next line.

---

## Notes

---

---

## Part II

# **ADVANCED EDITING PROCEDURES**

### **In this Section**

---

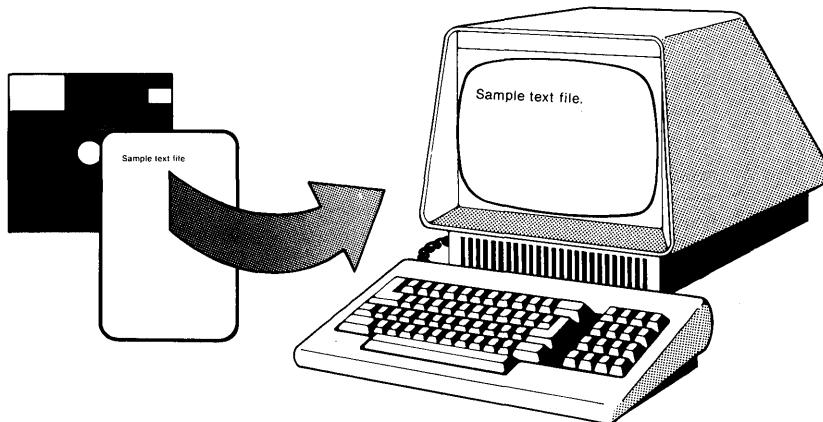
- Editing Multiple Files
- Split-Screen Editing
- Copying & Moving Text
- Searching

## Notes

## Chapter IX EDITING MULTIPLE FILES

### Introduction

When you wish to edit a document you have previously stored as a file, Perfect Writer retrieves a copy from disk storage and places it into a temporary workspace, called an 'editing buffer.'†



*Figure 4: Console screen is a window on your text in buffer.*

In this illustration a file document called 'LETTER.MSS' has been copied into an editing buffer. From the buffer it is being displayed onto the console screen.

Using such a procedure is convenient because changes that you make to the copy do not affect the original version of your document, which remains safely stored on disk. Thus, you can ultimately decide against keeping whatever changes you make and know that your original version is unaltered.

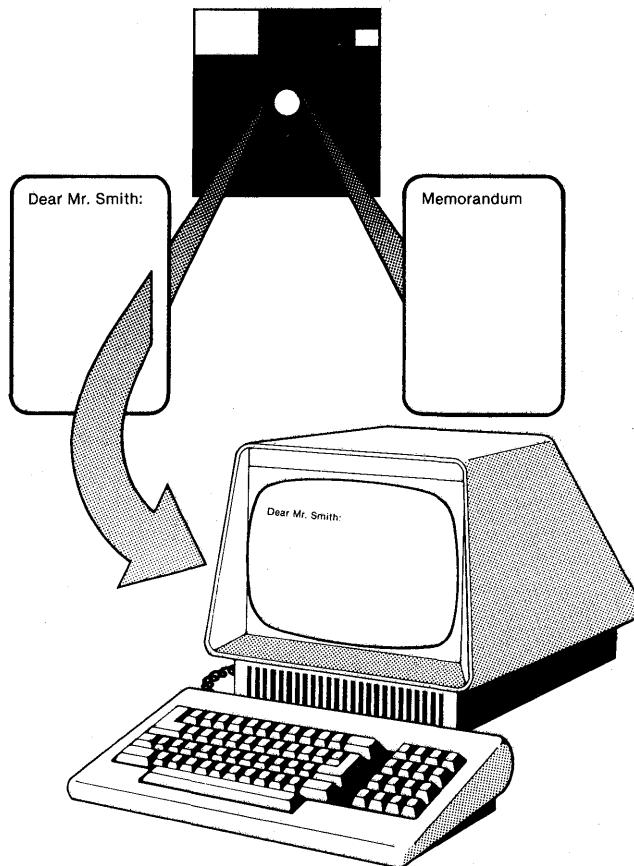
When you have finished editing, you can either replace the original version with the new, or store the new version in a separate file by itself. This, then, is the basic operation of editing a document.

† **Buffer** is a term used to define the space in the computer's memory where text material is temporarily stored while the computer is on.

---

## **Editing Several Files**

It is possible to have more than one document at a time in temporary editing buffers, as the following diagram illustrates:



*Figure 5: Console screen provides a separate window on two different documents.*

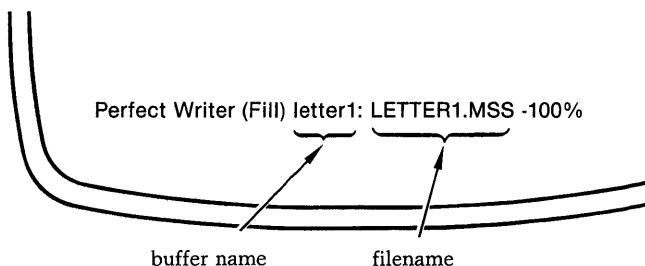
In this illustration, two files have been copied into editing buffers from disk storage. One is being displayed on the console screen, while the other is waiting to be displayed.

Up to seven documents can be held at one time in temporary editing buffers. As we shall see, the use of multiple buffers allows easy comparison and editing of documents, and especially easy transfer of text between documents.

## Buffer Names

Every editing buffer temporarily assumes a name when in use. This name is the name of the last file **from** which or **to** which a document has been copied. If the document in the buffer is new and has not yet been named, the default name 'NAME.ME' is assigned to the buffer.

The buffer takes as its name the first component of the filename. For example, if the file in the buffer is called 'LETTER1.MSS,' the buffer name will be 'LETTER1.' In the Mode Line both the buffer and file names will be displayed, thus:



*Figure 6: Mode Line with buffer names*

## **File & Buffer Commands**

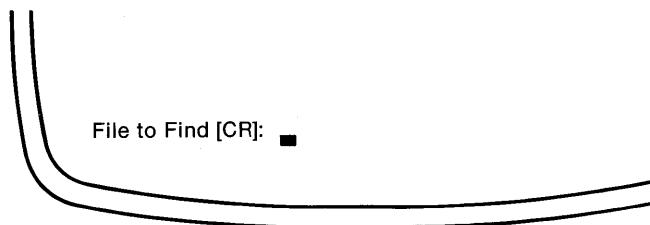
The following commands allow you to create buffers, delete buffers, copy documents into and out of buffers, switch the viewing screen from one buffer to another, and to list the names of buffers currently in use. You have already learned some of the file manipulation commands in the section on storing, and these commands round out the repertoire of commands you can use to create and manipulate files.

### **The FIND FILE Command**

Retrieves a stored document and places it in a buffer for you to edit. If the document you request is already in a buffer, this command switches you from the buffer currently being used to the buffer containing the document you requested. At any time during editing type the FIND FILE command:



Perfect Writer responds with the message:



Enter the name of the file followed by a carriage return. Perfect Writer retrieves the file and places it in a buffer for you to view.

**Note:** In retrieving a file from storage Perfect Writer always attempts to place the file in a buffer which has the same name. If you request a file whose name is associated with a buffer that is already in use, Perfect Writer will display the message: "Buffer in Use! Name of Buffer to Use?[CR]:". This circumstance often arises when requesting a second 'original' copy of a document you have been editing, in order to compare the changes you have been making.

For example, you have been editing a file document named 'LETTER1.MSS' in a buffer called 'LETTER1.' You wish to examine the original version of the document, which, of course, is also called 'LETTER1.MSS.' Using the FIND FILE command you get the message "Buffer in Use! Name of Buffer to Use?[CR]:"

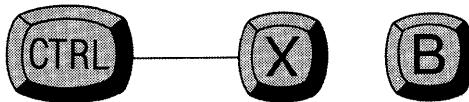
Perfect Writer in retrieving a second copy of the original 'LETTER1.MSS' has attempted to place it in a buffer which it wants to name 'LETTER1'—only to find that such a buffer name already exists!

When this situation arises, you must supply a different buffer name in which to place the second copy of the document. If you answer with only a carriage return, Perfect Writer will use the old buffer (and destroy anything that was in it). If you want to keep the old buffer, then you might enter: 'LETTER1a' as a new buffer name.

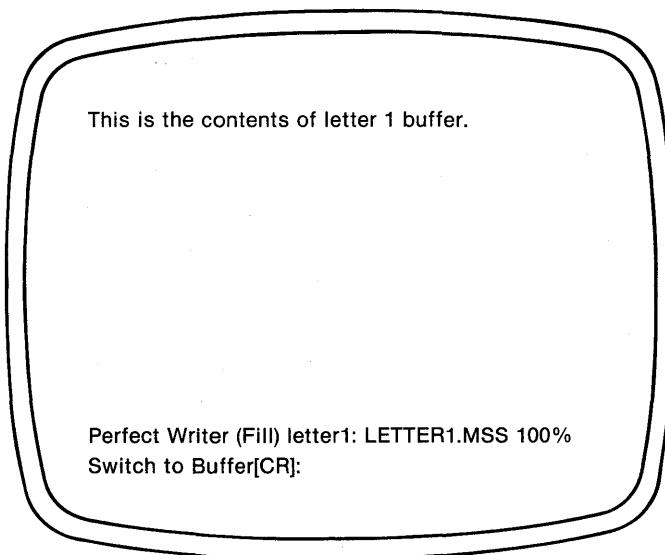
**Further Note:** The FIND FILE command is useful in creating new files. If the file you request does not exist, Perfect Writer will create a new file in an empty buffer, whose name will be the first component of the filename you have been given.

## The SWITCH BUFFERS Command

Allows you to switch between buffers actively in use, or to create a new blank buffer for entering a new document. Type the SWITCH BUFFERS command:



Perfect Writer will respond in the Echo Line with:



Enter the name of the buffer you wish to use, followed by a carriage return. (Remember: the buffer name is only the first portion of the filename.) Perfect Writer switches you to the buffer you have named. For example, if you enter: memo[CR] Perfect Writer will switch you to the buffer named "memo."

This is the contents of the memo buffer which has just been switched to.

Perfect Writer (Fill) memo: MEMO.MSS 100%

**Note:** If the buffer you name does not exist, Perfect Writer will respond with the message:

Create new buffer?[CR]: ■

"Y" causes Perfect Writer to create a new empty buffer identified by the name you have given, while "N" cancels the original command.

**Further Note:** The **filename** for any new buffer will be the default filename, 'NAME.ME,' and will remain such until you assign it another name, usually when 'writing the file' (see Chapter VI, page 7).

**One More Note:** The SWITCH BUFFERS command can be used to switch back and forth quickly between two buffers, without having to specify the buffer names each time.

If only a carriage return is entered when the message "Switch to Buffer[CR]:" is displayed, Perfect Writer will switch to the buffer that was last switched **from**.

## The BUFFERS DIRECTORY Command

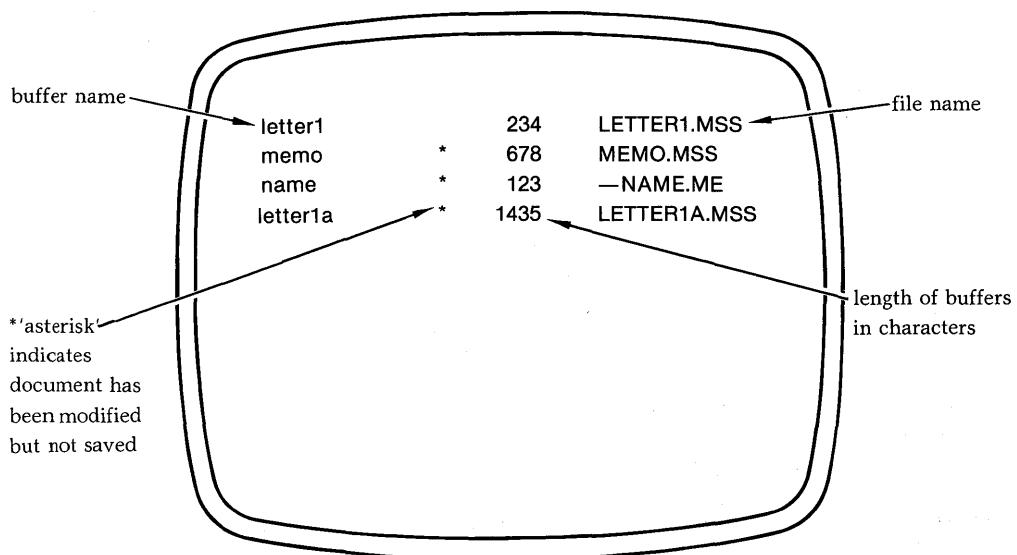
Instructs Perfect Writer to display a directory of all the buffers created during an editing session, together with the names and lengths of the documents they contain, and whether or not these documents have been 'saved.' This command is useful should you forget the names of the buffers that currently exist.

### Steps:

1. Type the BUFFERS DIRECTORY command:



Perfect Writer displays in the top left portion of the screen the list of current buffers. For example:



*Figure 7: Illustration of buffer directory*

The buffer directory will disappear when you resume editing.

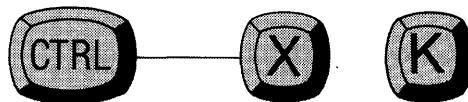
---

## The DELETE BUFFER Command

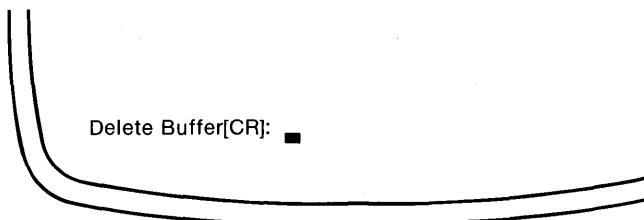
Deletes a buffer from the set of buffers actively being used. This command is useful in freeing storage space should a "Swap File Full" error message occur (see Appendix C, page 1.)

### Steps:

1. Type the DELETE BUFFER command:



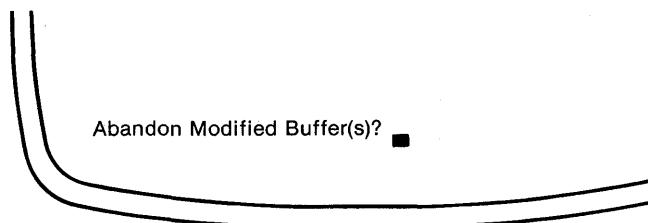
Perfect Writer will respond with the message:



2. Type the name of the buffer to be deleted.

## Quitting with Multiple Files

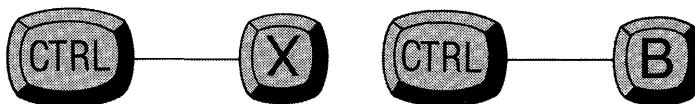
When you have finished editing multiple documents and wish to leave the "Writing/Editing" mode of Perfect Writer, typing the QUIT command (Control ----- X, Control ----- C) will, if you have not yet stored or saved any of the documents in active buffers, result in the following message from Perfect Writer:



A "yes" will erase all of the modifications you have just made to the current file including any modifications to other files in other buffers. "No" will instruct Perfect Writer to cancel the QUIT command and return you to what you were doing before you issued the QUIT command (giving you the opportunity to save your newly edited or created documents).

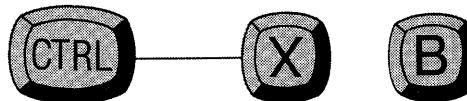
To save the documents existing in the various buffers:

1. Enter the BUFFERS DIRECTORY command:



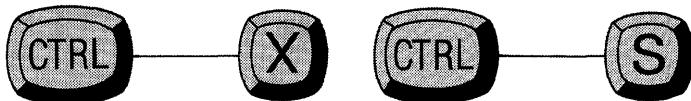
An asterisk, \*, beside a buffer name indicates that the document contained in the buffer has not been saved since it was last modified.

2. Switch to the first buffer whose document you wish to save, using the SWITCH BUFFERS command:



3. Save the document by giving the SAVE FILE command or the WRITE FILE command, depending on whether you wish to overwrite the old file or whether you wish to create a new file.

SAVE FILE:



WRITE FILE:



4. Continue switching to various buffers in this way, saving the documents that you wish.
5. Finally, give the QUIT command again, this time 'abandoning' the documents that you did not wish to save.



## Notes

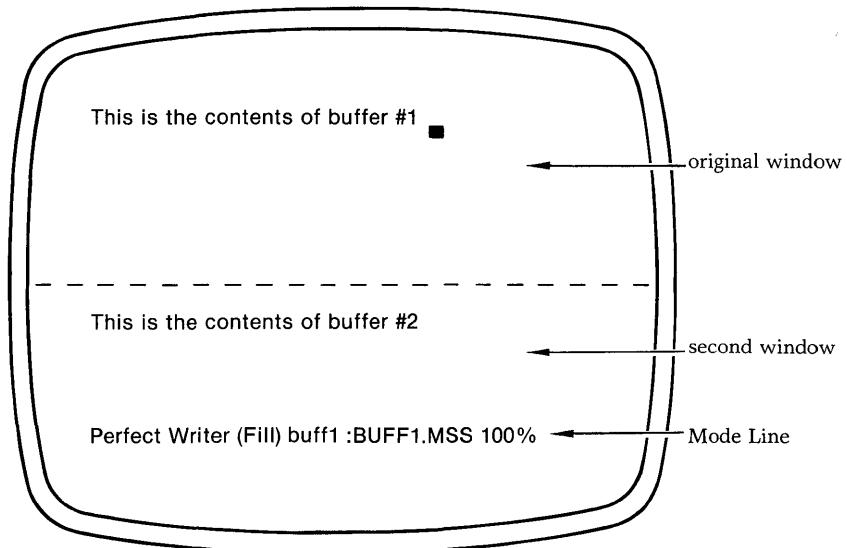
## Chapter X SPLIT-SCREEN EDITING

### Introduction

Perfect Writer offers the unique feature of allowing you to split the screen, or window, appearing on the console into two separate screens, or windows, one atop the other. Using two screens, it is extremely easy to modify one document based upon information in another, or to transfer sections of text from one document to another.

### Two Windows

Normally, the single screen window occupies almost all of the screen. In the split-screen option each window occupies a little less than half of the screen, the windows being separated by a row of dashes.



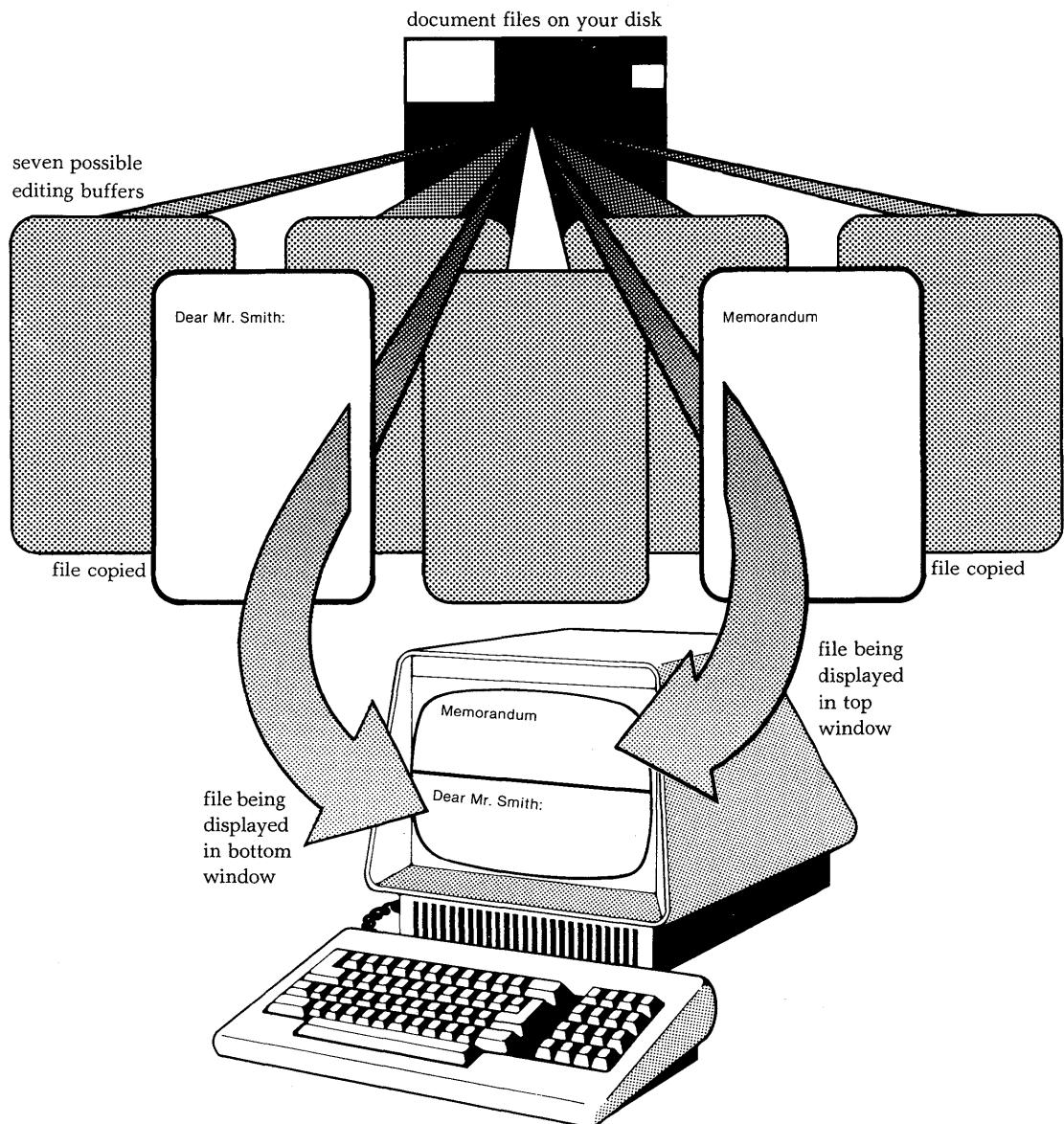
*Figure 8: Illustration of split screen*

When a second window is created it simply displays an identical copy of the document which was being edited in the original (full-screen) window. The original full-screen window, which still holds the cursor, has become the top window.

Initially both windows will display the same document belonging to the same buffer. Thus, editing changes made in one window will affect the other window also. However, the purpose of split-screen editing is that each window should display a different document held in a separate buffer. When this is the case, editing commands given in one window affect only the document being viewed in that window.

In the illustration to the right, two documents, 'LETTER.MSS' and 'MEMO.MSS,' have been retrieved from disk storage and placed in separate editing buffers, named 'letter' and 'memo' respectively. From these buffers both documents are being viewed on the screen simultaneously. The editing buffers each contain the full contents of their respective files. It should be understood that what are being viewed in the two windows are not just different documents, but **the contents of different editing buffers**. This is important because it means that you can view and edit simultaneously different portions of the **same** document. How is this accomplished? — with the TWO WINDOW command.

The following diagram illustrates the use of two windows:



*Figure 9: Console screen provides a separate window on two different documents.*

---

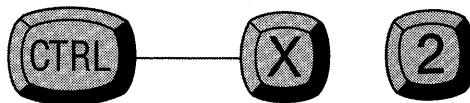
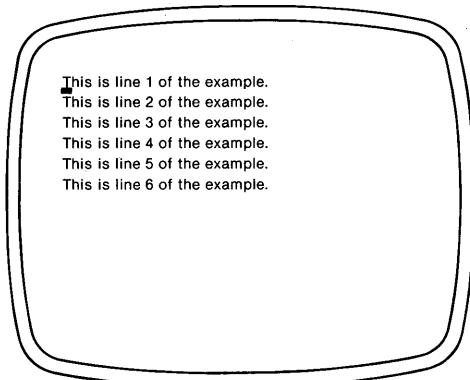
## WINDOW COMMANDS

The following commands govern the creation and manipulation of two windows:

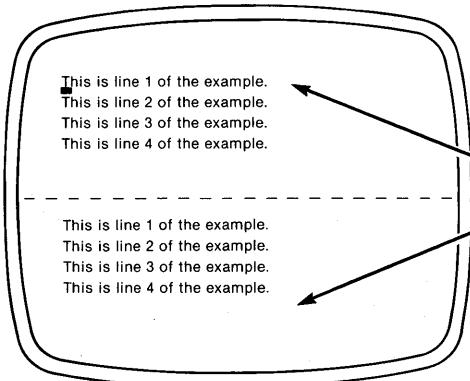
### The TWO WINDOWS Command

Splits the screen, changing one window into two and leaving the cursor in the upper window.

Before



After



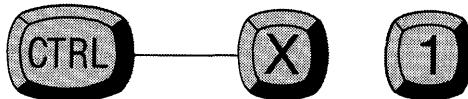
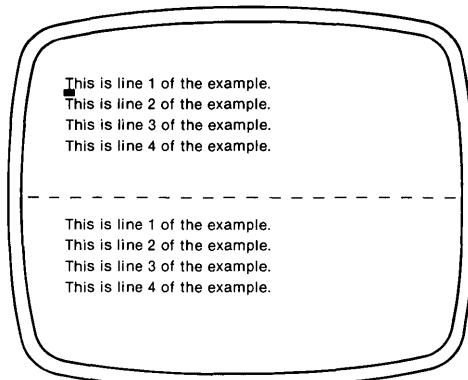
**Note:** Initially the text displayed in the lower (second) window will be identical to that displayed in the top window. This is because both windows look into the same editing buffer. Editing changes in one window will, therefore, affect the other window as well.

---

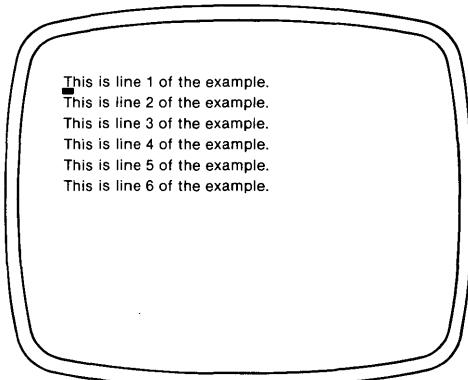
## The ONE WINDOW Command

Ends split screen editing by making whichever window the cursor is occupying the only window on the screen.

Before



After



## The ENLARGE WINDOW Command

Increases the size of the window holding the cursor by one line (thus **decreasing** by one line the size of the 'other window.')

Type the ENLARGE WINDOW command:

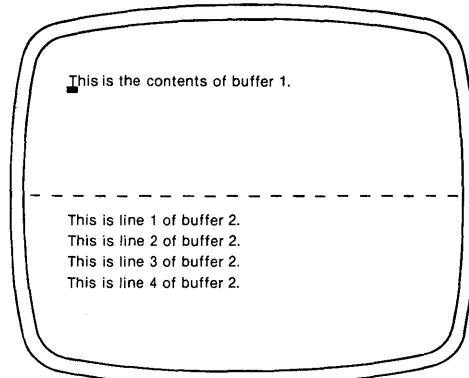


**Note:** A window may not be decreased in size to display fewer than three lines.

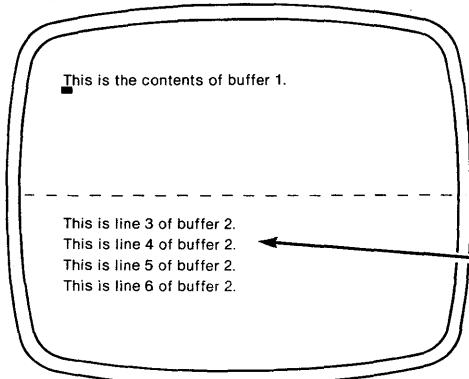
## The VIEW NEXT SCREEN (Other Window) Command

Displays the next screen of the window which the cursor is **not** occupying.

Before



After

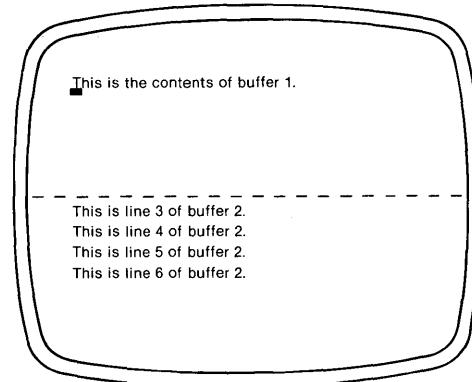


Note how  
text has  
scrolled up.

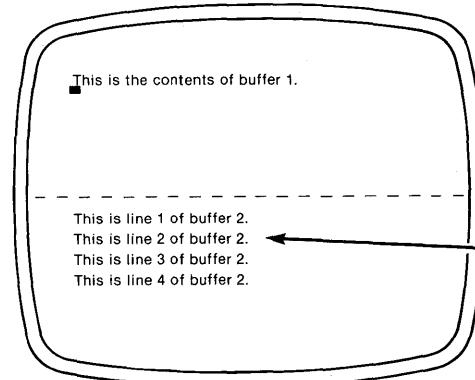
## The **VIEW PREVIOUS SCREEN (Other Window)** Command

Displays the previous screen of the window which the cursor is **not** occupying.

Before



After

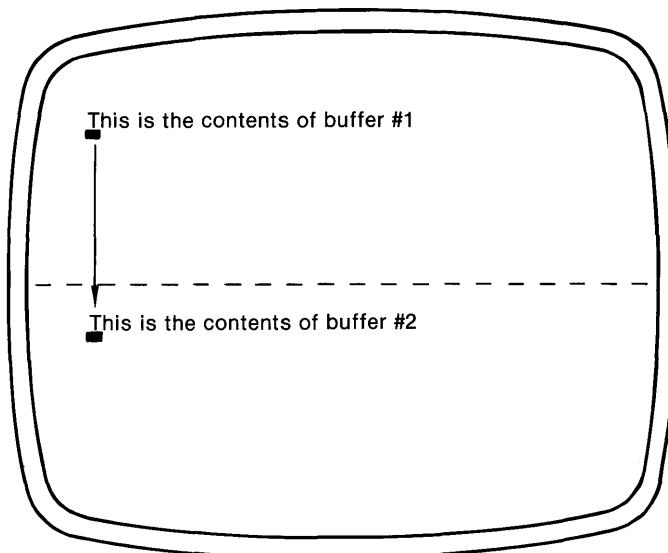


Note how  
text has  
scrolled down.

## The OTHER WINDOW Command



Switches the cursor from one window to the other.



**Note:** The Mode Line will change to reflect the buffer and filename of the new window.

---

**Exercise: Manipulating Two Windows**

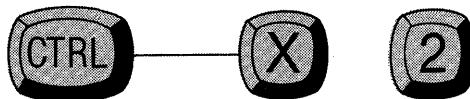
1. Call up Perfect Writer's exercise text "EDITME1.MSS" using the READ FILE command:



followed by:

EDITME1.MSS [CR]

2. Create two windows, by giving the TWO WINDOWS command:



The cursor will be in the top window, indicating that you are now editing in that window.

3. Call into the top window Perfect Writer's exercise text 'EDITME2.MSS,' using the FIND FILE command:



"EDITME2" will appear in the top window.

4. Practice switching back and forth between the two windows now, by giving the OTHER WINDOW command:



Notice the following:

- The Mode Line reflects the window that the cursor is in by displaying the correct document filename.
- When entering a window, the cursor returns to the position that it had occupied previously when in that window.

5. Practice changing the screen in the window which the cursor is **not** occupying, i.e. the 'other window,' using the VIEW NEXT SCREEN (OTHER WINDOW) command and the VIEW PREVIOUS SCREEN (OTHER WINDOW) command:

VIEW NEXT SCREEN:



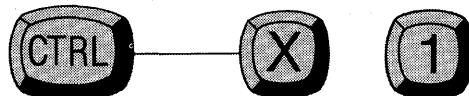
VIEW PREVIOUS SCREEN:



6. Enlarge one window, using the ENLARGE WINDOW command:

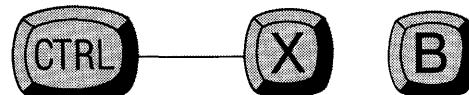


7. After you have enlarged one window (and necessarily decreased the other window), switch windows and reverse the process.
8. Finally, end the exercise by creating one window again, using the ONE WINDOW command:



The window which the cursor is occupying will become the only window on the screen.

**Note:** The document in the window that disappeared is still present in another editing buffer. You can easily switch to it giving the SWITCH BUFFER command.



## Chapter XI **COPYING & MOVING TEXT**

### **Introduction**

Perfect Writer provides extremely simple procedures for copying and moving portions of text. Whether text is to be shifted within a single document or between documents, the steps are essentially the same:

- Copy the word, line, or region of text into the 'temporary save buffer,' using an appropriate deletion command.
- Move the cursor to where the material is to be inserted.
- Restore the material from the temporary save buffer using the YANKBACK command.

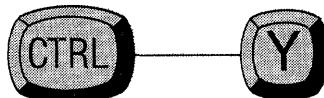
## Moving Text WITHIN a Document

### Steps:

1. Remove the text to be shifted from its current position using any one of the following deletion commands:
  - **DELETE PREVIOUS WORD** (Escape.....[Delete Key])
  - **DELETE NEXT WORD** (Escape.....D)
  - **DELETE LINE** (Control ---- K)
  - **DELETE ENTIRE LINE** (Escape.....Control ---- K)
  - **WIPE REGION procedure** (which includes):
    1. **MARK BOUNDARIES procedure** (see Chapter V, page 7) or  
**MARK WHOLE PARAGRAPH** (Escape.....H)
    2. **WIPE REGION** (Control ---- W)

Perfect Writer automatically saves the deleted material in the temporary save buffer.

2. Move the cursor to the new location in the document where the material is to be placed.
3. Restore the deleted material using the YANKBACK command:



**Exercise:**

Call up Perfect Writer's exercise text, "EDITME3.MSS," by typing:



followed by:

"editme3.mss" [CR]

On this text practice shifting lines and paragraphs:

1. Move the first sentence to just after the paragraph.
2. Move the paragraph to the beginning of the text.
3. Switch sentences two and four.

**EDITME3.MSS: Practicing Moving Text**

This is sentence ONE; it goes after the paragraph.

This is sentence TWO; it will get switched.

This is sentence THREE; it will stay put.

This is sentence FOUR; where does it go?

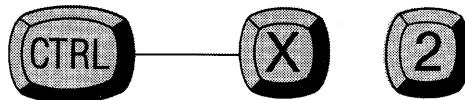
This is the last paragraph, which is going to be placed at the beginning of the text. That will be a trick. Actually, it is simple: Mark the boundaries, 'Wipe' the region, move the cursor, and 'Yank' it back!

---

## Moving Text BETWEEN Documents

### Steps:

1. Create two windows by giving the TWO WINDOWS command:



2. In one window call up the second document you wish to transfer material **from**, using the FIND FILE command:



3. Remove the text to be shifted using one of the following deletion commands:

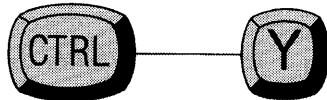
- **DELETE PREVIOUS WORD** (Escape.....[Delete Key])
- **DELETE NEXT WORD** (Escape.....D)
- **DELETE LINE** (Control ---- K)
- **DELETE ENTIRE LINE** (Escape.....Control ---- K)
- **DELETE SENTENCE FORWARD** (Escape.....K)
- **WIPE REGION procedure** (which includes):
  1. **MARK BOUNDARIES procedure** (see Chapter V, page 7) or  
**MARK WHOLE PARAGRAPH** (Escape.....H)
  2. **WIPE REGION** (Control ---- W)

Perfect Writer automatically saves the deleted material in the temporary save buffer.

4. Switch to the other window using the OTHER WINDOW command:



5. Move the cursor to the location where the material is to be inserted.
6. Restore the material from the temporary save buffer using the YANKBACK command:

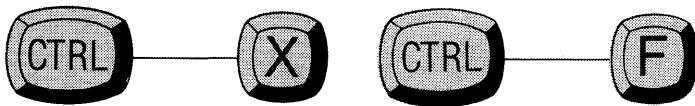


**Exercise:**

Practice shifting text between documents by moving a paragraph between two of Perfect Writer's exercise texts, "EDITME1.MSS" and "EDITME2.MSS."

**Steps:**

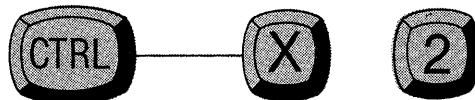
1. Call up Perfect Writer's exercise text "EDITME1" by typing the FIND FILE command:



followed by:

editme1.mss [CR]

2. Create two windows, using the TWO WINDOWS command:



3. In the window you are in (the top window) call up Perfect Writer's exercise text "EDITME2" by typing the FIND FILE command (as in step 1).
4. Mark the boundaries of the first paragraph in "EDITME2" using the MARK BOUNDARIES procedure (see Chapter V, pages 7-8), or the MARK WHOLE PARAGRAPH command:



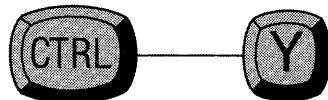
5. Delete the paragraph into the temporary save buffer, using the COPY REGION command:



6. Switch to the other window using the OTHER WINDOW command:



7. Position the cursor at the end of the text "EDITME1."
8. Recall the deleted text from the temporary save buffer using the YANKBACK command:



## The COPY REGION Command

When moving text it is often desirable to keep the material to be moved in its original location, even though you wish to copy it to another location as well. This can be accomplished in two ways:

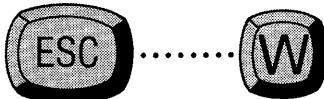
1. By restoring the text in its original location before moving the cursor to the new location. (Once saved in the temporary save buffer, deleted material can be recalled by the YANKBACK command ANY number of times.)
2. By using the COPY REGION command:

### Steps:

1. Mark the region to be copied using the MARK BOUNDARIES procedure (see Chapter V, pages 7-8), or the MARK WHOLE PARAGRAPH command:

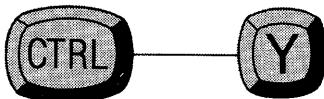


2. Copy the region to be moved using the COPY REGION command.



Nothing will appear to have happened except that a plus sign, '+', will appear at the right of the Mode Line, indicating that the region has been copied into the temporary save buffer.

3. Move the cursor to the location where the material is to be inserted.
4. Restore the material from the temporary save buffer using the YANKBACK command:



## GATHERING AND MOVING TEXT

Although deletions are normally accumulated in the temporary save buffer only one group at a time (see discussion, Chapter V, page 6), it is possible to move throughout your text, copying or deleting words, sentences, and paragraphs, and then to replace them as a group at a single location elsewhere.

In order to gather text in this way Perfect Writer must be instructed to **continue saving** previous text deletions. This is accomplished by using the CONTINUE SAVING command. The following steps illustrate this gathering process:

### Steps:

1. Delete or copy into the temporary save buffer the first sentence, paragraph, or region of text to be gathered and moved, using any of the deletion or copy commands:

- **DELETE PREVIOUS WORD** (Escape.....[Delete Key])
- **DELETE NEXT WORD** (Escape.....D)
- **DELETE LINE** (Control ---- K)
- **DELETE ENTIRE LINE** (Escape.....Control ---- K)
- **DELETE SENTENCE FORWARD** (Escape.....K)
- **WIPE REGION procedure** (which includes):
  1. **MARK BOUNDARIES procedure** or **MARK WHOLE PARAGRAPH** (Escape.....H)
  2. **WIPE REGION** (Control ---- W)
- **COPY REGION** (Escape.....W)

The plus-sign, '+', will appear in the Mode Line indicating that your deletion has been saved in the save buffer.

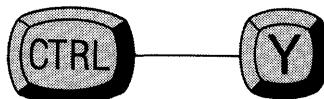
2. Move the cursor to the next deletion.

3. Your first deletion is now in danger of being lost from temporary storage! Repositioning the cursor caused Perfect Writer to stop accumulating deletions. Perfect Writer now expects you to restore your first deletion using the YANKBACK command. (Note that the '+' sign in the Mode Line has disappeared.) Before making the next deletion you must tell Perfect Writer to continue saving the first deletion. To do this type the **CONTINUE SAVING** command:



The '+' sign which had disappeared from the Mode Line has now reappeared, indicating that your previous deletion will be saved along with the deletion you are about to make.

4. Delete or copy into the 'temporary save buffer' the next word, sentence, or region of text to be gathered and moved using any of the deletion or copy commands (see above).
5. In this way continue moving about the text gathering deletions into the temporary save buffer. **REMEMBER: Always give the CONTINUE SAVING command immediately BEFORE making a deletion!**
6. Position the cursor at the location where you wish the gathered material to be inserted.
7. Recall all the deletions you have made from the temporary save buffer using the YANKBACK command:



**Note:** The newly grouped material will probably be merged together. You may have to insert spaces between sentences, or add new lines, or even paragraph indentations. As you practice gathering and shifting text, you will see that this can be accomplished during the deleting process, by appending spaces, indentations, or 'newline' characters (all invisible) to your text before deleting it.

---

## Notes

## Chapter XII **SEARCHING**

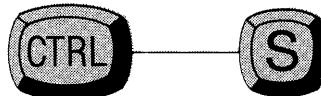
Several commands are provided that instruct Perfect Writer to search through a document for an item. Depending upon the command, Perfect Writer will position the cursor at the item, or will replace the item with another.

### The FORWARD SEARCH Command

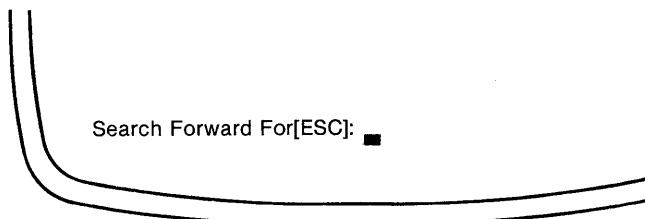
Instructs Perfect Writer to search forward from the present position of the cursor to the first occurrence of an identified item, and to place the cursor just **after** the item.

#### Steps:

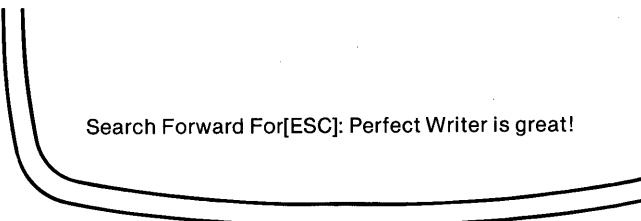
1. Type the FORWARD SEARCH command:



Perfect Writer will respond with the message:



2. Enter the characters, word, or string of words you wish to search for. For example:



3. Type the Escape key to begin the search.

Perfect Writer searches forward from the position of the cursor to the first occurrence of "Perfect Writer is great!", placing the cursor after "great!"

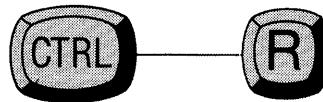
**Note:** To search for successive occurrences of an item, continue to type the search command: Control ---- S, Escape. It is not necessary to retype the search string. Perfect Writer will continue to search for the last named item.

## The REVERSE SEARCH Command

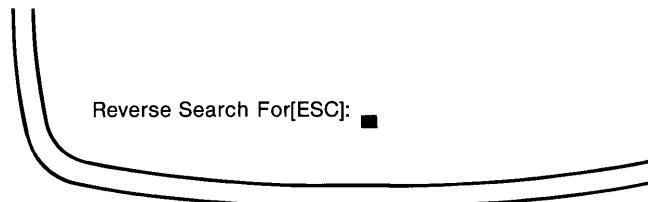
Instructs Perfect Writer to search backward from the present position of the cursor to the first occurrence of an identified item and to place the cursor just **before** the item.

### Steps:

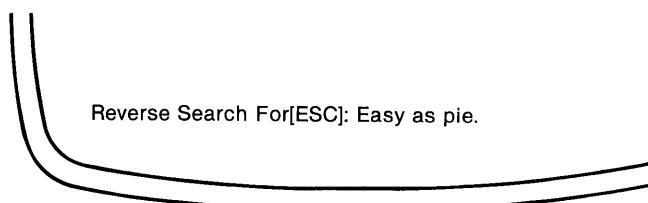
1. Type the REVERSE SEARCH command:



Perfect Writer will respond with the message:



2. Enter the characters, word, or string of words you wish to search for. For example:



3. Type the Escape key to begin the search. Perfect Writer searches backward from the position of the cursor to the first occurrence of "Easy as pie," placing the cursor before "Easy."

**Note:** To search for successive occurrences of an item, continue to type the search command: Control ---- R, Escape. It is not necessary to retype the search string. Perfect Writer will continue to search for the last named item.

---

## The SEARCH AND REPLACE Command

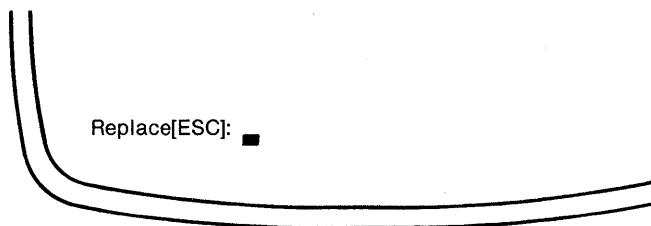
Causes Perfect Writer to search from the position of the cursor to the end of the text for all occurrences of a character, word, or words, and to replace them with another character, word, or words.

### Steps:

1. Position the cursor at the beginning of the text and type the SEARCH AND REPLACE command:



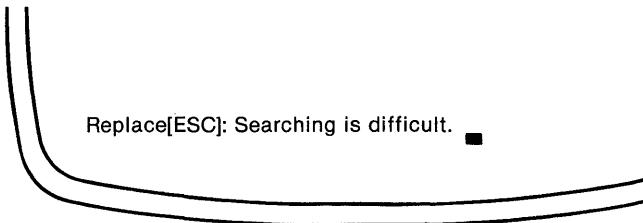
Perfect Writer responds in the Echo Line:



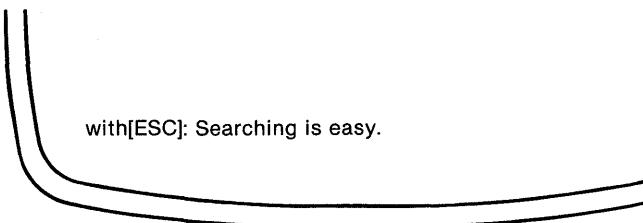
2. Then enter the word to replace:



For example, if you type the SEARCH AND REPLACE command and enter,  
"Searching is difficult"



and then, "Searching is easy"



Perfect Writer will search for every occurrence of the word string "Searching is difficult" and replace it with "Searching is easy."

---

## The SEARCH AND REPLACE (With Query) Command

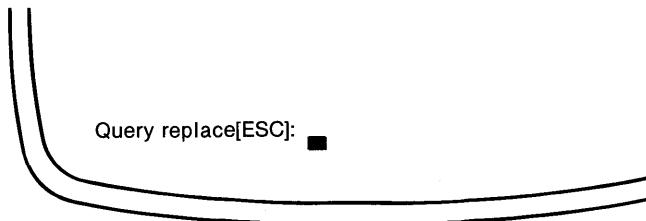
Causes Perfect Writer to search for and replace specified characters or words, beginning at the position of the cursor and continuing to the end of the text. However, at each occurrence of the item to be replaced, Perfect Writer pauses to ask for confirmation of your intent to replace.

### Steps:

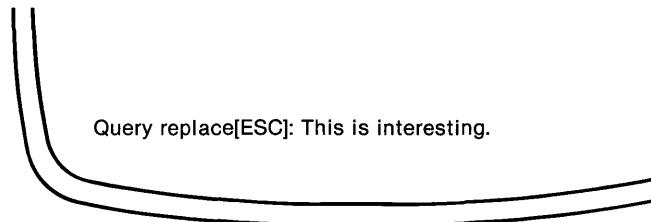
1. Position the cursor at the beginning of the text and type the SEARCH AND REPLACE (With Query) Command:



Perfect Writer responds with:



2. Enter the characters or words to be replaced, followed by the Escape key. For example:



Perfect Writer will respond:

with [ESC]:

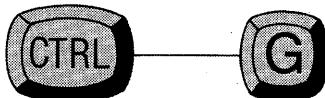
3. Enter the words to be substituted, followed by the Escape key. For example:

with[ESC]: This is marvelous.

Perfect Writer will find the first occurrence of "This is interesting." Before replacing it with "This is marvelous!" it will display the following message in the Echo Line:

Replacing 'This is interesting.' with 'This is marvelous.' Okay?

4. At this point you may select one of several options:

<b>Option</b>	<b>Perfect Writer will:</b>
You may:	
1. Give the CANCEL command: 	... Cease searching and replacing, leaving the cursor where it was when the CANCEL command was given.
2. Type a 'period': 	... Cease searching and replacing, but return the cursor to the original position where the search command was given.
3. Type an exclamation character: 	... Replace all remaining occurrences of the item without stopping each time to ask for confirmation.
4. Type a comma: 	... Replaces the old item with the new, and then asks if you are satisfied with the result: "Confirm Replace?" ("Yes" causes Perfect Writer to continue to the next item; "No" causes the original item to be restored and the search continued.)
5. Type 'Y', 'y', or (space): 	... Replace the present item and continue the search.
6. Type (any other key):	... Will not replace the present item, but will continue the search for the next item.

## Not Found

If an item is not found, the message "Not found" will appear at the left of the Echo Line. However, the cursor will be left after the closest match it could find.

## Uppercase vs. Lowercase

A lowercase letter in the word or string of words to be searched will find and match either a lowercase letter or an uppercase letter in the text. However, an **uppercase letter will find and match only an uppercase letter.** Example:

**coffee** will find and match **coffee** and **COFFEE**  
**COFFEE** will find and match only **COFFEE**

## Mismatches

Perfect Writer searches for and matches strings of **characters**. Instructed to search for the word 'the' in the text, it will find and match such words as: '**thesis**', '**'theatre**', and '**father**'. To eliminate such mismatching enclose the word to be searched in 'blanks,' using the space bar, for example:

(space)**the**(space)

Though a blank space is invisible, Perfect Writer nevertheless recognizes it as a character.

---

## Notes

## Notes

## Notes

---

---

## Part III

# **DOCUMENT DESIGN**

---

### **In this Section:**

- Overview
  - Environment Format Commands
  - List Format Commands
  - Typeface Commands
  - Document Sectioning and Organization
  - Tools for Form Letter Design
  - Document Style Commands
  - Lessons
-

---

## ENVIRONMENT COMMANDS

ADDRESS	XIV—8	INDENT	XIV—25
CENTER	XIV—7	ITEMIZE	XIV—18
CLOSING	XIV—8	LEVEL	XIV—12
DESCRIPTION	XIV—20	QUOTATION	XIV—6
DISPLAY	XIV—26	TEXT	XIV—4
ENUMERATE	XIV—16	UNDELT	XIV—24
EXAMPLE	XIV—3	VERBATIM	XIV—2
FLUSHLEFT	XIV—10	VERSE	XIV—22
FLUSHRIGHT	XIV—10		

## TYPEFACE COMMANDS

BOLDFACE	XV—1	SUBSCRIPT	XV—4
BOLD ITALIC	XV—3	SUPERSCRIPT	XV—4
ITALIC	XV—3	TYPEWRITER	XV—5
ONE-WORD	XV—5	UNDERSCORE	XV—2
ROMAN	XV—5		

## DOCUMENT SECTIONING AND ORGANIZATION

<b>UNNUMBERED HEADINGS</b>		<b>NUMBERED HEADINGS</b>	
UNNUMBERED	XVI—2	CHAPTER	XVI—4
MAJORHEADING	XVI—2	SECTION	XVI—4
HEADING	XVI—2	SUBSECTION	XVI—4
SUBHEADING	XVI—3	PARAGRAPH	XVI—4
		APPENDIX	XVI—5
		APPENDIXSECTION	XVI—5

## ORGANIZATION

INDEX	XVI—9
FOOT	XVI—10
NOTE	XVI—11
PAGEHEADING	XVI—12
PAGEFOOTING	XVI—12

## TOOLS FOR FORM LETTER DESIGN

BLANKPAGE	XVII—8	NEWPAGE	XVII—10
BLANKSPACE	XVII—9	REF	XVII—16
CASE	XVII—1	SET	XVII—16
COMMENT	XVII—7	STRING	XVII—11
INCLUDE	XVII—4	TITLE	XVII—15
MESSAGE	XVII—5	VALUE	XVII—12

---

## **Chapter XIII OVERVIEW**

In this section of the manual you will learn about the document design capabilities of Perfect Writer. As you will see Perfect Writer radically changes the conventional approach to typing and preparing manuscripts. With Perfect Writer you no longer have to worry about setting tabs and margins and line spacing. There is no need to devise special formats for lists, quotations, or other special texts. Instead, Perfect Writer allows you to choose from a wide variety of custom format options that automatically print your document the way you want.

Included in the document design and printing procedures of Perfect Writer are such features as automatic paging, creation of a table of contents, an alphabetized index, page headings, footnotes, page footings, and much more. All of these features are fully integrated with Perfect Writer's editing functions. If you decide to add three new paragraphs to your document and to delete a footnote, then simply use the Perfect Writer editing commands to make these changes. While reformatting your text, Perfect Writer automatically rennumbers the pages and footnotes to accommodate what changes you make. Your new document will be printed exactly as you have designed it.

## No Need to Modify Commands

Once format commands have been placed in your document, Perfect Writer is able to adapt those commands to the capabilities of any printer.

For example, if your command specifies that a word be italicized, but the printer you are using can only underline, Perfect Writer directs the printer to do the best it can, i.e. **underline**. However, should you later acquire a printer with italicizing capabilities, Perfect Writer will recognize this and direct your new printer to **italicize** the word properly—all on the authority of your original command!

## Greater Editing Freedom

Once you have specified a format for a portion of text, you need care no longer how that segment of text appears on your screen. Perfect Writer will rearrange the text to match the format you have specified.

As you will see, the effect of this is to free you of the burden of having constantly to oversee the myriad details of formatting your document (i.e., adjusting margins, tabbing, spacing, indenting, etc.).

## **Format Commands**

Document design format commands are embedded in the text of your document at the time of writing and editing, and before the document is stored in a file. Every format command indicates two things:

1. The format option to be used.
2. The portion of text to be formatted.

### **The '@' (At-sign) & 'Fences'**

All format commands begin with the @ (at-sign), while the text to be formatted is normally enclosed within 'fences.' Fences can be any of the following:

(...), [...], {...}, <...>, "...",

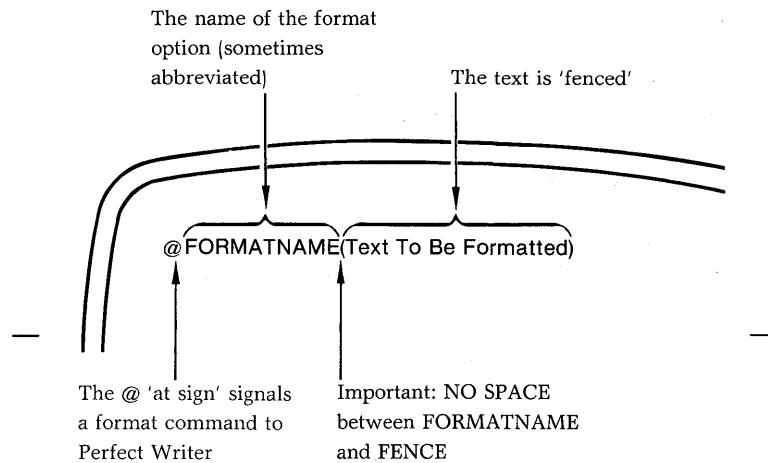
and '...'

Thus to format a paragraph as a quotation simply place the '@QUOTATION' format name at the beginning of the paragraph and place any of the above fences around the paragraph to be formatted as a quotation. In a sense, you are fencing in the material you want formatted.

---

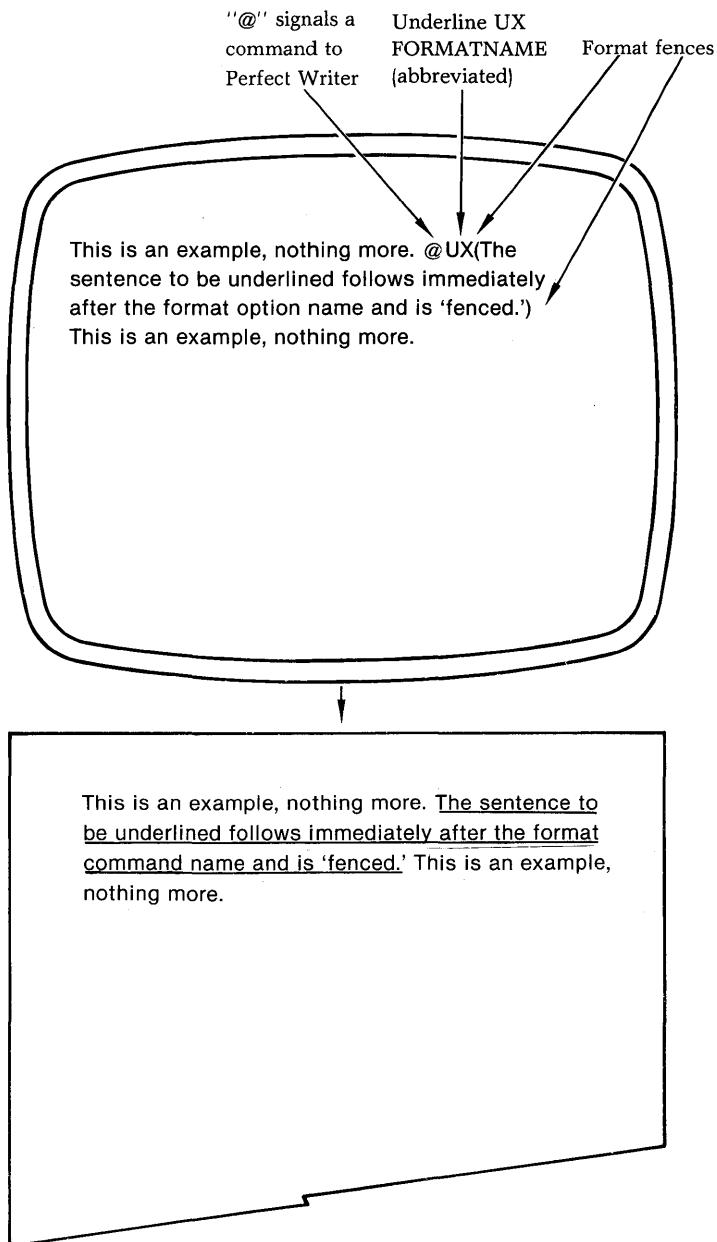
## General Form of the Format Command

In general, a format command follows the form:



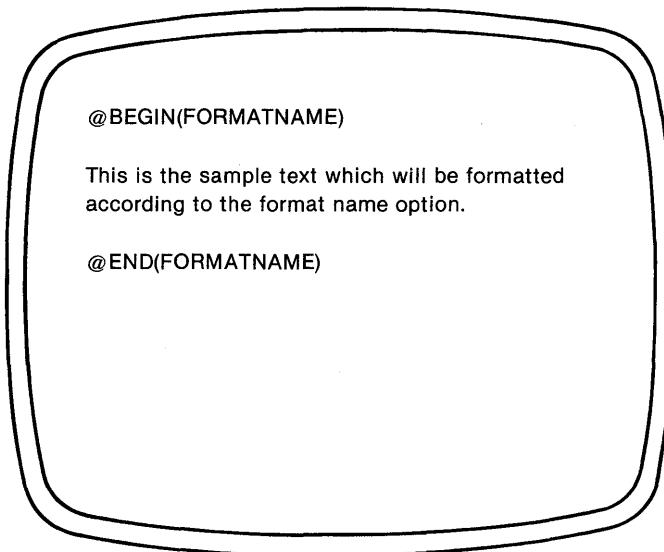
### Example:

You wish to underline a sentence in a paragraph. The UNDERLINE command is placed in the following manner:



## BEGIN-END Command Form

If the text to be formatted is long and contains within it other nested format commands, it is often useful to use the BEGIN-END command:

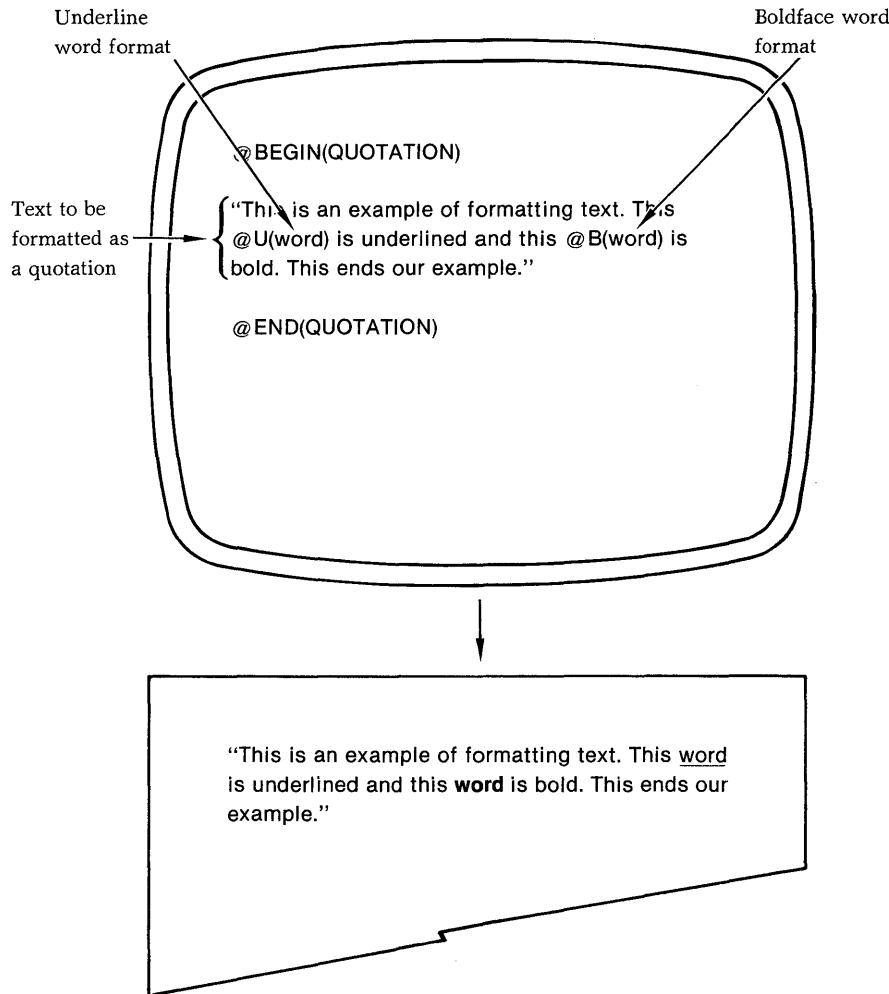


Note that in this command form the name of the **format option is enclosed in fences**, while the text to be formatted is not. This can only be used with the environment commands in Chapter XIV.

---

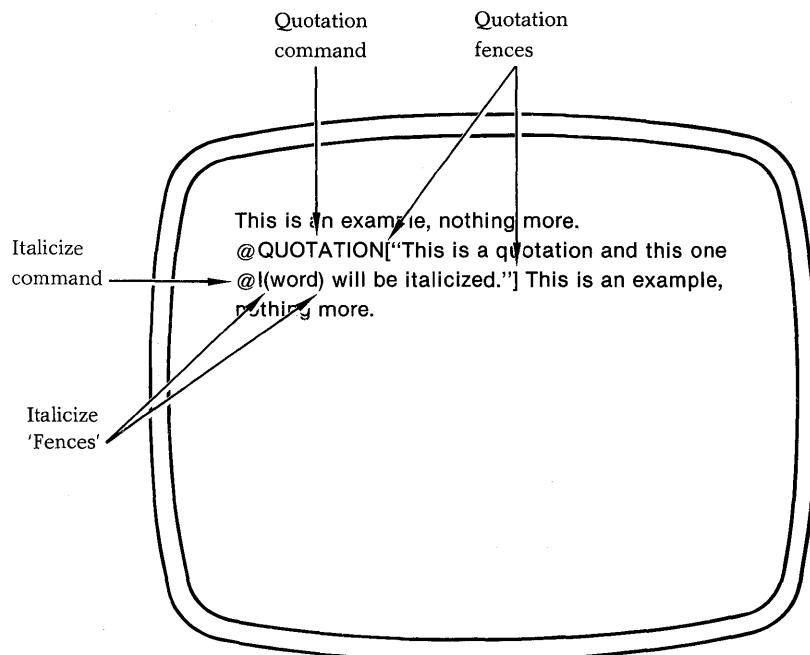
**Example:**

You wish to format as a quotation several lines (or even paragraphs) of text, in which are nested other format commands. Place the format commands as follows:



## 'Nesting' Format Commands

Formatting commands can be placed one within another in a process called 'nesting,' as in the following example, in which a word within a quotation is underlined:



Note that in the nesting example different fences for the two commands were used, i.e. brackets, [ ], for the quotation, and parentheses, ( ) for the underlined word. This was to avoid confusion. When nesting formatting commands it is extremely important not to confuse or omit any fences. Missing fences or similar pairs of fences enclosed one within another will result in an error message from Perfect Writer.

---

## **Sequence of Format Commands**

Generally, format commands are inserted in the text in the place where formatting is to occur.

However, format commands that deal with the document as a whole, or with portions of text that repeat throughout the document, such as PAGEHEADING commands or style parameter commands that determine margins, tabs, etc., are placed at the very beginning of the document before all other text.

## **Style**

Many of the document design formats available can be altered using style parameter commands, which govern indentation, margins, spacing, and so on. Where a format option can be altered using a style parameter, it will be so noted in that format options list of 'characteristics.' Style parameter commands and their use are discussed in Chapter XVIII.

## **Formatting Errors**

From time to time you will improperly place a formatting command—omit a fence, fail to supply a numbered dimension to a style parameter, etc. This will cause Perfect Writer to generate an error message identifying the cause of the problem. In most cases, Perfect Writer will specify the line on which the error occurred.

The error messages that Perfect Writer may issue are listed in Appendix B, together with a simple explanation of what they mean.

## Notes

## Chapter XIV

# **ENVIRONMENT FORMAT COMMANDS**

This chapter presents the 'Environment Format' commands, those format options that apply to bodies of text, i.e. lists, quotations, examples, etc. Each format option is provided with a list of features, followed by the command form and an illustration of the output that can be expected.

## @VERBATIM

### The VERBATIM Environment

VERBATIM is the normal (or default) print mode of Perfect Writer. VERBATIM causes all text to be reproduced exactly as it appears on the screen.

#### Features:

- Lines remain as they are.
- Tabs and blankspaces are preserved.

---

#### The Command Form:

@BEGIN(VERBATIM)

This is a

free form text to illustrate  
the use of the VERBATIM format.

Everything is  
printed exactly as you create it on the  
screen.

Nothing is justified to make it fit or  
look better....

@END(VERBATIM)

---

#### The Result:

This is a sample paragraph. This is a sample paragraph.  
This is a sample paragraph. This is a sample paragraph. This is  
a sample paragraph. This is a sample paragraph.

This is a  
free form text to illustrate  
the use of the VERBATIM format.  
Everything is  
printed exactly as you create it on the  
screen.

Nothing is justified to make it fit or  
look better....

This is a sample paragraph. This is a sample paragraph.  
This is a sample paragraph. This is a sample paragraph. This is  
a sample paragraph. This is a sample paragraph.

## **@EXAMPLE**

### **The EXAMPLE Environment**

Provides a simple format for offsetting examples and other illustrative matter from the rest of the text.

#### **Features:**

- Long lines are 'wrapped' but short lines are left unchanged (i.e. they are not 'filled').
- Text is single-spaced and indented one-half inch on the left and one-quarter inch on the right.
- Blankspace and tabs in the text are left unchanged.

---

#### **The Command Form:**

**@EXAMPLE**(This illustrates the EXAMPLE format.

Notice that short lines  
are left unchanged,  
while long lines that do not fit are 'wrapped' to the next line.  
The text is inset by one-half inch on the left  
and one-quarter inch on the right.)

---

#### **The Result:**

This is a sample paragraph. This is a sample paragraph.  
This is a sample paragraph. This is a sample paragraph. This is  
a sample paragraph. This is a sample paragraph.

This illustrates the EXAMPLE format option.  
Notice that short lines  
are left unchanged,  
while long lines that do not fit are 'wrapped' to  
the next line.  
The text is inset by one-half inch on the left  
and one-quarter inch on the right.

This is a sample paragraph. This is a sample paragraph.  
This is a sample paragraph. This is a sample paragraph. This is  
a sample paragraph. This is a sample paragraph.

## @TEXT

### The TEXT Environment

'Cleans-up' a text by grouping and indenting all paragraphs, justifying margins, wrapping and filling all lines. This format option can be used to override the normal VERBATIM print mode.

#### Features:

- Long lines are 'wrapped' and short lines are 'filled.'
  - Lines will be 'justified' according to the value of the 'justification' style parameter (default = yes).
  - Line spacing, paragraph indentations, and spacing between paragraphs are determined by 'spacing,' 'indent,' and 'spread' style parameters respectively.
-

**The Command Form:**

**@BEGIN(Text)**

This is a sample document which originally had ragged margins, unfilled lines, and no justification. Some paragraphs were indented, some were not.

This format can be used to override the VERBATIM print mode by simply enclosing the entire document within a TEXT format—i.e. place @@BEGIN(TEXT) at the beginning of the document, and @@END(TEXT) at the end of the document.

**@END(TEXT)**

---

**The Result:**

This is a sample document which originally had ragged margins, unfilled lines, and no justification. Some paragraphs were indented, some were not.

This format can be used to override the VERBATIM print mode by simply enclosing the entire document within a TEXT format—i.e. place @BEGIN(TEXT) at the beginning of the document, and @END(TEXT) at the end of the document.

---

Note that in order to print an '@' sign you need to type it twice, otherwise Perfect Writer will execute any command following the '@' sign.

## **@QUOTATION**

### **The QUOTATION Environment**

Provides a format for quotations and other parenthetical remarks.

#### **Features:**

- Long lines are 'wrapped' and short lines 'filled'; paragraphs are indented.
- Lines are single-spaced and inset one-half inch from either margin.
- The right edge of the text is either justified or left ragged, depending upon the value of the 'justification' style parameter.
- Paragraph indentations and spacing between paragraphs are determined by the value of the 'indent' and 'spread' style parameters respectively.

---

#### **The Command Form:**

**@QUOTATION("This is a quotation. It is single spaced and indented on both sides one-half inch. All lines have been 'wrapped' and 'filled.'**

Notice that you have to add the quotation marks yourself."  
—Howard H. Wade, 1982.)

---

#### **The Result:**

This is a sample paragraph. This is a sample paragraph.  
This is a sample paragraph. This is a sample paragraph. This is  
a sample paragraph. This is a sample paragraph.

"This is a quotation. It is single-spaced  
and indented on both sides one-half inch.  
All lines have been 'wrapped' and 'filled.'

Notice that you have to add the quotation  
marks yourself." —Howard H. Wade, 1982.

This is a sample paragraph. This is a sample paragraph.  
This is a sample paragraph. This is a sample paragraph. This is  
a sample paragraph. This is a sample paragraph.

## **@CENTER**

### **The CENTER Environment**

Causes all lines within a text to be centered between the left and right margins.

#### **Features:**

- Lines are neither 'wrapped' nor 'filled.'
- A line too long to fit between the margins will begin at the left margin and extend beyond the right margin.
- Tabs and blankspaces in the text remain unchanged.

---

#### **The Format Command:**

**@CENTER**(This is an example of the CENTER format.  
Each line  
is  
centered.)

---

#### **The Result:**

This is a sample paragraph. This is a sample paragraph.  
This is a sample paragraph. This is a sample paragraph. This is  
a sample paragraph. This is a sample paragraph.

This is an example of the CENTER format.  
Each line  
is  
centered.

This is a sample paragraph. This is a sample paragraph.  
This is a sample paragraph. This is a sample paragraph. This is  
a sample paragraph. This is a sample paragraph.

**@ADDRESS**

**@CLOSING**

## The ADDRESS & CLOSING Environments

Causes the return address and the closing salutation to be properly positioned in letters. Essentially these commands are identical. They are also useful in creating a 'column' of text on the right side of a page.

### Features:

- Lines are neither 'wrapped' nor 'filled'; thus, line lengths remain unchanged.
  - The text is **left** justified, with the left edge beginning in the center of the page.
  - Tabs and blankspace in the text remain unchanged.
-

**The Command Form:**

@ADDRESS(Wall Street Software  
71 Murray Street  
New York, NY 10007)

Dear \_\_\_\_\_

This is a sample paragraph. This is a sample paragraph.  
This is a sample paragraph. This is a sample paragraph. This  
is a sample paragraph. This is a sample paragraph.

@CLOSING(Yours very truly,

Jennifer Kaufman)

**The Result:**

<p>Wall Street Software 71 Murray Street New York, NY 10007</p> <p>Dear _____</p> <p>This is a sample paragraph. This is a sample paragraph. This is a sample paragraph. This is a sample paragraph. This is a sample paragraph. This is a sample paragraph.</p> <p>Yours very truly,</p> <p>Jennifer Kaufman</p>
---

**@FLUSHLEFT**

**@FLUSHRIGHT**

### **The FLUSH Environments**

FLUSHLEFT causes the lines of a text to begin at the left margin.

FLUSHRIGHT causes all lines of text to end against the right margin.

#### **Features:**

- Lines are neither 'wrapped' nor 'filled'; thus, line lengths remain the same.
  - All tabs and blankspaces in the text are preserved.
-

**The @FLUSHLEFT Command Form:**

@FLUSHLEFT(This is an example of FLUSHLEFT.  
Each line begins at the left margin,  
except that  
    tabs and blank         spaces and  
line-breaks  
are preserved as they were in the original.)

**The @FLUSHRIGHT Command Form:**

@FLUSHRIGHT(This is an example of FLUSHRIGHT.  
Each line ends against the right margin. Notice that  
line lengths and  
blank spaces         are the same as in the  
original.)

**The Result:**

This is a sample paragraph. This is a sample paragraph.  
This is a sample paragraph. This is a sample paragraph. This is a sample paragraph. This is a sample paragraph.

This is an example of FLUSHLEFT.  
Each line begins at the left margin,  
except that  
    tabs and blank         spaces and  
line-breaks  
are preserved as they were in the original.

This is an example of FLUSHRIGHT.  
Each line ends against the right margin. Notice that  
line lengths and  
blank spaces         are the same as in the  
original.

This is a sample paragraph. This is a sample paragraph.  
This is a sample paragraph.

## LIST ENVIRONMENT COMMANDS

### @LEVEL

#### The LEVEL Environment

Causes a series of items or paragraphs to be numbered. LEVEL formats can be 'nested' one within another.

#### Features:

- Long lines are 'wrapped' and short lines are 'filled.'
  - The right edge is either justified or left ragged depending upon the value of the 'justification' style parameter.
  - Spacing between paragraphs is determined by the value of the 'spread' style parameter.
  - Paragraphs can be 'undented' or not, depending upon the 'levelhang' style parameter (default is **No** or not undented).  
If undented, the number will 'stick-out' from the block of the paragraph. If not undented, the number will be within the block of the paragraph.
  - 'Nested' paragraphs can be indented or not, depending upon the value of the 'levelindent' style parameter (see Chapter XVIII, page 4). If not indented, paragraphs will differ only in their numbering.  
If indented, nested paragraphs will be inset by one-half inch (this is the default).
-

**The Command Form:**

(These examples show two 'nested' LEVEL formats, in which the paragraphs are first undented, and then not undented.

@BEGIN(LEVEL)

This is the FIRST numbered paragraph....

This is the SECOND numbered paragraph....

This is the THIRD numbered paragraph....

@LEVEL(Here begins the FIRST nested paragraph....

Here begins the SECOND nested paragraph....)

} Nested level  
command

This is the FOURTH numbered paragraph....

This is the FIFTH numbered paragraph....

@END(LEVEL)

---

Note: The blank lines between sentences separate items.

---

**The Result:**

**@STYLE(levelhang no) default  
(NOT UNDENTED)**

This is a sample paragraph. This is a sample paragraph.  
This is a sample paragraph. This is a sample paragraph. This is  
a sample paragraph. This is a sample paragraph. This is a  
sample paragraph. This is a sample paragraph. This is a  
sample paragraph. This is a sample paragraph.

1. This is the FIRST numbered paragraph .....

.....

2. This is the SECOND numbered paragraph .....

.....

3. This is the THIRD numbered paragraph .....

.....

3.1. Here begins the FIRST nested paragraph .....

.....

3.2. Here begins the SECOND nested paragraph .....

.....

.....

4. This is the FOURTH numbered paragraph .....

.....

5. This is the FIFTH numbered paragraph .....

.....

This is a sample paragraph. This is a sample paragraph.  
This is a sample paragraph. This is a sample paragraph. This is  
a sample paragraph. This is a sample paragraph. This is a  
sample paragraph. This is a sample paragraph. This is a  
sample paragraph. This is a sample paragraph.

---

Note that the numbers are included in the bodies of the paragraph.

**@STYLE{levelhang yes}  
(UNDEDENTED)**

This is a sample paragraph. This is a sample paragraph.  
This is a sample paragraph. This is a sample paragraph. This is a  
sample paragraph. This is a sample paragraph. This is a  
sample paragraph. This is a sample paragraph. This is a  
sample paragraph. This is a sampe paragraph.

1. This is the FIRST numbered paragraph .....
2. This is the SECOND numbered paragraph .....
3. This is the THIRD numbered paragraph .....
- 3.1. Here begins the FIRST nested paragraph .....
- .....
- 3.2. Here begins the SECOND nested paragraph .....
- .....
4. This is the FOURTH numbered paragraph .....
5. This is the FIFTH numbered paragraph .....
- .....

This is a sample paragraph. This is a sample paragraph.  
This is a sample paragraph. This is a sample paragraph. This is  
a sample paragraph. This is a sample paragraph. This is a  
sample paragraph. This is a sample paragraph. This is a  
sample paragraph. This is a sample paragraph.

Note that the numbers are **not** included in the bodies of the paragraph.

## **@ENUMERATE**

### **The ENUMERATE Environment**

Causes items or paragraphs to be numbered and inset from the document margins. ENUMERATE formats can be 'nested.'

#### **Features:**

- Long lines are 'wrapped' and short lines are 'filled.'
  - Numbers are indented two characters and the paragraph itself six characters.
  - The right edge of the text is either justified or left ragged depending upon the setting of the 'justification' style parameter (default = yes).
  - Spacing between paragraphs is determined by the value of the 'spread' style parameter (default = 1 line).
-

### **The Command Form:**

(This example shows two 'nested' ENUMERATE commands. Although single items have been used, whole paragraphs could have been used as well.)

@BEGIN(ENUMERATE)  
Sculpture

Paintings  
@ENUMERATE(Oil

Water

Acrylic)  
Photography

Clothing  
@END(ENUMERATE)

---

### **The Result:**

This is a sample paragraph, nothing more. This is a sample paragraph, nothing more.

1. sculpture
2. paintings
  1. oil
  2. water
  3. acrylic
3. photography
4. clothing

---

Note that blank lines are used to separate the items.

## **@ITEMIZE**

### **The ITEMIZE Environment**

Causes items in a list to be offset from the document margins and marked with dashes or asterisks.

#### **Features:**

- All lines are 'wrapped' and 'filled.' Thus, original line lengths are not preserved.
  - The right edge of the text is either justified or left ragged depending upon the value of the 'justification' style parameter.
  - Spacing between paragraphs is determined by the value of the 'spread' style parameter.
  - Dashes are indented from the margins by two spaces, and the paragraph or item is indented by five.
-

### **The Command Form:**

This example shows two 'nested' itemize commands. Though single items have been used to illustrate this command, paragraphs could have been used as well.

```
@BEGIN(ITEMIZE)
Sculpture
```

```
Paintings
@ITEMIZE(Oil
```

```
Water
```

```
Acrylic)
Photography
```

```
Clothing
@END(ITEMIZE)
```

---

### **The Result:**

This is a sample paragraph, nothing more. This is a sample paragraph, nothing more.

- sculpture
- paintings
  - \* oil
  - \* water
  - \* acrylic
- photography
- clothing

This is a sample paragraph, nothing more. This is a sample paragraph, nothing more. This is a sample paragraph, nothing more. This is a sample paragraph, nothing more.

---

Note that blank lines are used to separate the items.

## **@DESCRIPTION**

### **The DESCRIPTION Environment**

Causes an item to be separated and displayed away from the paragraph which describes it.

#### **Features:**

- Lines are single-spaced, 'wrapped' and 'filled.' Thus, original line lengths are not preserved.
  - The right edge of the text is either justified or left ragged depending upon the 'justification' style parameter.
  - Spacing between paragraphs is determined by the 'spread' style parameter.
  - Paragraphs are indented approximately one-fourth of the distance between the left and right margins.
-

### The Command Form:

**@BEGIN(DESCRIPTION)**

apple@\\ a red fruit that is very useful in making apple pies.  
Apples have a slight crunchy texture.

orange@\\ a round orange-colored fruit used in making  
orange juice.

watermelon@\\ a large oblong melon with lots of seeds—a  
summertime favorite.

**@END(DESCRIPTION)**

**Note:** The “@ ” (at-sign, slash) occurring just after each item serves as a ‘tab’ which causes the remaining description to be set back and grouped into a paragraph. This ‘tab’ can be used without a descriptive item. (On some terminals there is no back slash. If your terminal doesn’t have a back slash, try a normal slash ‘/’.) Also, blank lines are used to separate descriptions.

---

### The Result:

apple	a red fruit that is very useful in making apple pies. Apples have a light crunchy texture.
orange	a round orange fruit used in making orange juice.
watermelon	a large oblong fruit with lots of seeds—a summertime favorite.

## **@VERSE**

### **The VERSE Environment**

Provides a means of formatting verse as well as other unnumbered and unmarked lists of items.

#### **Features:**

- Long lines are 'wrapped' but short lines are left unchanged (i.e. they are not 'filled').
  - The text as a whole is indented, while paragraphs within the text are 'undented.'
  - Tabs and blankspaces within the text are not preserved, but replaced with single spaces.
  - The right margin is inset 10 characters.
-

(Example 1)

**The Command Form:**

@BEGIN(VERSE)

Roses are red,

Violets are blue,

This line is too long to fit on a single line and will 'wrap,'

Which is a very handy thing for it to do!

@END(VERSE)

---

(Example 2)

**The Command Form:**

@BEGIN(VERSE)

These two paragraphs illustrate the other use of the VERSE format, that of listing unnumbered and unmarked items, such as paragraphs.

Notice that both of these paragraphs are indented from the document margins just as a 'verse' would be. However, their first lines are 'undented' for easy identification.

@END(VERSE)

---

**The Result:**

Roses are red  
violets are blue,  
This line is too long to fit on a single line  
and will 'wrap,'  
Which is a very handy thing for it to do!

These two paragraphs illustrate the other use of the VERSE format, that of listing unnumbered and unmarked items, such as paragraphs.

Notice that both of these paragraphs are indented from the document margins just as a 'verse' would be. However, their first lines are 'undented' for easy identification.

## @UNDENT

### The UNDENT Environment

Creates paragraphs whose first lines are 'undented.'

#### Features:

- Long lines are 'wrapped' and short lines 'filled.'
- The right edge of the text is either justified or not depending upon the value of the 'justification' style parameter (default is justified).
- Spacing between paragraphs is determined by the value of the 'spread' style parameter (default is one line).
- How far the body of the paragraph is indented from the first line (or how far the first line is undented from the paragraph) is determined by the value of the 'indent' style parameter (default is two characters).

---

#### The Command Form:

@UNDENT(This illustrates the UNDENT format, in which the first line of each paragraph is 'undented' from the body of the paragraph.

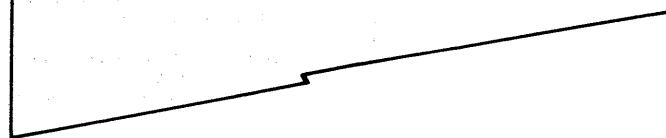
How far the first line is undented is determined by the value of the 'indent' style parameter—an easy thing to change.)

---

#### The Result:

This illustrates the UNDENT format, in which the first line of each paragraph is 'undented' from the body of the paragraph.

How far the first line is undented is determined by the value of the 'indent' style parameter—an easy thing to change.



## **@INDENT**

### **The INDENT Environment**

Causes Perfect Writer to indent all text by one-half inch. Like the DISPLAY command, the INDENT command sets off text for highlighting. Only the right margin is set.

---

#### **The Command Form:**

**@INDENT**(This is an example of the INDENT command.  
Notice that the right hand margin is not affected by Perfect  
Writer.)

---

#### **The Result:**

This is a sample paragraph, nothing more. This is a sample paragraph, nothing more. This is a sample paragraph, nothing more. This is a sample paragraph, nothing more.

This is an example of the indent command. Notice that the right hand margin is not altered by Perfect Writer.

This is a sample paragraph, nothing more. This is a sample paragraph, nothing more. This is a sample paragraph, nothing more. This is a sample paragraph, nothing more.

## @DISPLAY

### The DISPLAY Environment

The left margin of all text is inset by one-half inch and the right margin is inset by one-quarter inch. This environment is useful for displaying examples or programs that should be offset slightly from the rest of the text. Also see the EXAMPLE, QUOTATION and INDENT environments.

---

#### The Command Form:

This is a sample paragraph, nothing more. This is a sample paragraph, nothing more.

@DISPLAY(This is the text being displayed. Notice how it is set off for display. Both the left and right margins are indented.)

This is a sample paragraph, nothing more. This is a sample paragraph, nothing more.

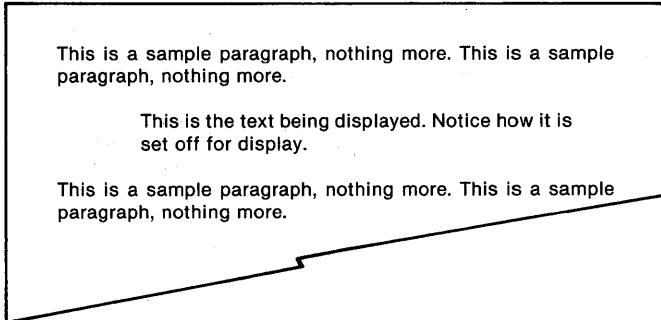
---

#### The Result:

This is a sample paragraph, nothing more. This is a sample paragraph, nothing more.

This is the text being displayed. Notice how it is set off for display.

This is a sample paragraph, nothing more. This is a sample paragraph, nothing more.



---

## Chapter XV

### TYPEFACE COMMANDS

If your printer allows for backspacing, underlining, and different type fonts, then you can use these with the Perfect Writer typeface format commands. As with other format commands, you simply place a fence around the text you want printed in a special font and precede the fence with a selected typeface format command.

#### **@B** The BOLDFACE Command

Causes text to appear in 'boldface' (darker than surrounding text).

---

#### The Command Form:

**@B(These words are in Boldface.)**

---

#### The Result:

This is a sample paragraph, nothing more. This is a sample paragraph, nothing more.

**These words are in Boldface.**

This is a sample paragraph, nothing more. This is a sample paragraph, nothing more.

Note: The BEGIN and END fence commands cannot be used with any of the format commands in this or any of the following chapters.

---

@U

@UN

@UX

## UNDERLINING Command

Perfect Writer offers three underlining options:

---

### @U 1) Command Form:

@U(All of the printing characters in this sentence are underlined, including punctuation...However, blank spaces are not underlined.)

#### The Result:

All of the printing characters in this sentence are underlined, including punctuation...However, blank spaces are not underlined.

---

### @UN 2) Command Form:

@UN(Just the alphanumeric characters are underlined in this sentence, not punctuation... or spaces!)

#### The Result:

Just the alphanumeric characters are underlined in this sentence, not punctuation...or spaces!

---

### @UX 3) Command Form:

@UX(Everything is underlined in this sentence—characters, spaces, and punctuation!)

#### The Result:

Everything is underlined in this sentence—characters, spaces, and punctuation!

---

**@I**

## The ITALICS Command

Causes text to be printed in italics. If the printing device is not capable of italicizing, underlining is used instead.

---

### The Command Form:

**@I**(This text is in italics, which produces a nice contrast to surrounding text.)

---

### The Result:

*This text is in italics, which produces a nice contrast to surrounding text.*

**@P**

## The BOLDFACE ITALICS Command

Causes text to be italicized and boldfaced. If the machine printer is not capable of italicizing, underlining is used instead.

---

### The Command Form:

**@P**(This line is in BOLDFACE ITALICS—What does it look like?)

---

### The Result:

*This line is in BOLDFACE ITALICS—What does it look like?*

---

---

@+ (**SUPERscript**)

@- (**SUBscript**)

### The SUPERscript and SUBscript Environments

Causes characters to be printed above or below the line of regular text. (These format options are **not** used for footnotes. See Chapter XVI, page 10.)

Extra lines above and below the line containing super- and subscripts can be added to prevent confusion of super- and subscripts on successive lines. This is accomplished by the 'scriptpush' style parameter (default = yes). (See Chapter XVIII, page 5.)

All format commands embedded within super- or subscript commands are ignored except @B (boldface), @R (roman-type), or @T (typewriter text).

---

#### The Command Form:

The first half of this sentence has a @ - (word) subscripted,  
while the second half has a @ + (word) superscripted.

---

#### The Result:

The first half of this sentence has a        subscripted, while  
    word  
    word  
the second half has a        superscripted.

---

**@W**

## The ONE-WORD Command

Causes material to be considered as **one** word. When printing Perfect Writer will not split the word(s) between lines or insert extra space within it while justifying.

---

### The Command Form:

Alla ka Zam! Abra Ca Dabra Doo! What does @W(Alla ka Zam! Abra Ca Dabra Doo?)

---

### The Result:

Alla ka Zam! Abra Ca Dabra Doo! What does  
Alla ka Zam! Abra Ca Dabra Doo?

**@R**

**@T**

## The ROMAN & TYPEWRITER Commands

Causes printing to resume in ordinary typewriter or roman characters. Useful for emphasizing a character or word within a 'boldface,' 'italics,' or 'boldface italics' format.

---

### The Command Form:

@P[All but two of these words are in boldface italics. Can you @R(spot) the @T(two) that are not?]

---

### The Result:

*All but two of these words are in boldface italics. Can you spot the two that are not?*

---

## **Notes**

## Chapter XVI

# DOCUMENT SECTIONING AND ORGANIZATION

This chapter begins an introduction to the sectioning and organizing design options offered by Perfect Writer. The commands presented here are used to section and title the parts of a document. Examples of both the command form and the resulting printer output have been provided whenever possible.

## **SECTIONING COMMANDS**

### **Headings**

Perfect Writer offers various styles of headings to title and section a document. Headings can be numbered or unnumbered, centered, underlined, boldfaced, or plain. Some headings cause Perfect Writer to begin a new page of text, while other headings are only preceded by several blank lines. Numbered headings automatically generate an entry to be included in a table of contents for the document. Following are the various heading formats available in Perfect Writer.

### **Unnumbered Headings**

Two basic unnumbered headings exist in Perfect Writer: MAJORHEADING and SUBHEADING.

## @MAJORHEADING

### The MAJORHEADING Format

#### Features:

- Causes a boldfaced, unnumbered, centered heading to be printed at the beginning of the text.
- Long lines of the heading are 'wrapped' but short lines are not 'filled.'
- Tabs and blankspaces in the heading are not preserved, i.e. they are replaced with a single blankspace.

---

Three choices for a MAJORHEADING are available:

**UNNUMBERED** Perfect Writer skips to a **new page**, leaving **six** blank lines before the heading and two after.

**Command Form:**

**@UNNUMBERED(This is a MAJORHEADING)**

**MAJORHEADING** Perfect Writer does not skip to a new page, but leaves **six** blank lines before the heading and two after.

**Command Form:**

**@MAJORHEADING(This is a MAJORHEADING)**

**HEADING** Perfect Writer leaves **four** blank lines before the heading and two after.

**Command Form:**

**@HEADING(This is a MAJORHEADING)**

---

#### The Result:

This is a MAJORHEADING

This is a sample paragraph, nothing more. This is a sample paragraph, nothing more. This is a sample paragraph, nothing more. This is a sample paragraph, nothing more.

## @SUBHEADING

### The SUBHEADING Format

Causes an underlined heading to be printed at the left margin.

#### Features:

- Long lines are 'wrapped,' but short lines are not 'filled.'
- Multiple blankspaces and tabs in the text are replaced with a single space.
- The subheading is preceded by **four** blank lines and followed by two.

---

#### Command Form:

**@SUBHEADING(This is a SUBHEADING)**

---

#### The Result:

This is a sample paragraph, nothing more. This is a sample paragraph, nothing more.

This is a SUBHEADING

This is a sample paragraph, nothing more. This is a sample paragraph, nothing more.

## NUMBERED SECTION HEADINGS/TABLE OF CONTENTS

### Introduction

It is possible to generate a table of contents for any document by using numbered headings to divide and section the document. Each numbered heading generates an entry in the table of contents, containing the section number, section title, and page on which the section or subsection starts.

Possible numbered headings are:

<b>@CHAPTER</b>	Begins a new page, leaves six blank lines, and prints a boldfaced, centered, numbered chapter heading at the beginning of the text, followed by three more blank lines. Also, initiates the numbering sequence for any subordinate sections that follow.
<b>@SECTION</b>	Prints an underlined, numbered heading flush with the left margin. This heading is preceded by four blank lines and followed by two.
<b>@SUBSECTION</b>	Prints a numbered heading, which is not underlined, flush with the left margin. This heading is preceded by two blank lines and followed by one.
<b>@PARAGRAPH</b>	Prints a numbered heading, which is not underlined, flush with the left margin. This heading is preceded by two blank lines and followed by one.

**@APPENDIX**

Like the CHAPTER command, 'APPENDIX' begins a new page, leaves six blank lines, and prints a bold-faced, centered, and numbered appendix heading at the head of the text, followed by three more blank lines. It initiates the numbering sequence for following subordinate APPENDIXSECTIONS.

**@APPENDIXSECTION**

Leaves four blank lines and prints an underlined appendix section heading flush with the left margin, followed by two blank lines.

---

**Example:**

The following sequence of numbered headings might represent a portion of a User Manual for Perfect Writer.

---

**Command Form:**

@CHAPTER (Basic Commands)  
@SECTION (Moving the Cursor)  
@SUBSECTION (The FORWARD CHARACTER command)  
@SUBSECTION (The BACKWARD CHARACTER command)  
@SECTION (Deleting)  
@SUBSECTION (The DELETE ENTIRE LINE command)  
@SUBSECTION (The WIPE REGION command)  
@PARAGRAPH (The SET MARK command)  
@APPENDIX (Installing Perfect Writer)  
@APPENDIXSECTION (Console)  
@APPENDIXSECTION (Printer)

---

**Results:**

Chapter 1 Basic Commands	
1.1 Moving the Cursor	.....
1.1.1 The FORWARD CHARACTER Command.	.....
1.1.2 The BACKWARD CHARACTER Command.	.....
1.2 Deleting	.....
1.2.1 The DELETE ENTIRE LINE Command.	.....
1.2.2 The WIPE REGION Procedure.	.....
1.2.2.1 The SET MARK Command.	.....
Appendix A Installing Perfect Writer	
A.1 Console	.....
A.2 Printer	.....

The table of contents for this outline would appear as follows:

Table of Contents†	
Chapter 1 Basic Commands	
1.1 Moving the Cursor	
1.1.1 The FORWARD CHARACTER Command	
1.1.2 The BACKWARD CHARACTER Command	
1.2 Deleting	
1.2.1 The DELETE ENTIRE LINE Command	
1.2.2 The WIPE REGION procedure	
1.2.2.1 The SET MARK Command	
Appendix A Installing Perfect Writer	
A.1 Console	
A.2 Printer	

†Note: When printed Perfect Writer would include page numbers.

**General Note:**

Main chapter and appendix headings will be numbered and formatted depending upon the value of the 'chapter' style parameter. If the 'chapter' style parameter is set to "yes," then chapters and appendix headings will be formatted and numbered (default is yes). If the parameter is "no" numbering will begin at the next lowest division which is 'Section.'

---

Section Numbering with  
@STYLE{chapters}

	Yes}	No)
CHAPTER	1.	N/A
SECTION	1.1	1.
SUBSECTION	1.1.1	1.1
PARAGRAPH	1.1.1.1	1.1.1
APPENDIX	A.	N/A
APPENDIXSECTION	A.1	1.

---

If the parameter is set to "no" in a document containing CHAPTER and APPENDIX heading commands, an error will result. The document will be printed, but the old chapter headings will be formatted awkwardly, since Perfect Writer doesn't know what to do with them.

---

## ORGANIZATION

### @INDEX

#### The INDEX Command

This command instructs Perfect Writer to generate an index for items that you tag throughout your text. The list of items will automatically be printed alphabetically at the end of the document together with the page numbers on which the item occurs.

---

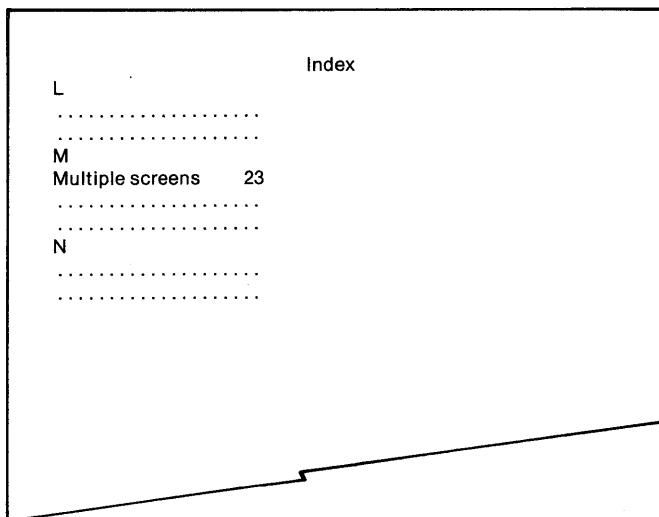
#### Command Form:

**@INDEX(multiple screens)** Perfect Writer allows the use of multiple screens to ease the editing process.

—23—

---

#### The Result:



## @FOOT

### The FOOTNOTE Command

Causes Perfect Writer to format the designated material as a footnote, inserting in the text a number to reference the footnote.

The value of 'footpush' style parameter (see Chapter XVIII, page 3) determines whether the reference number is printed as a 'superscript,' (default) or is enclosed in brackets, [ ].

Depending on the setting of the 'notes' style parameter (see Chapter XVII, page 3), the footnote will be placed at the bottom of the page, the end of the text, or within the line itself enclosed in brackets (default is bottom of the page).

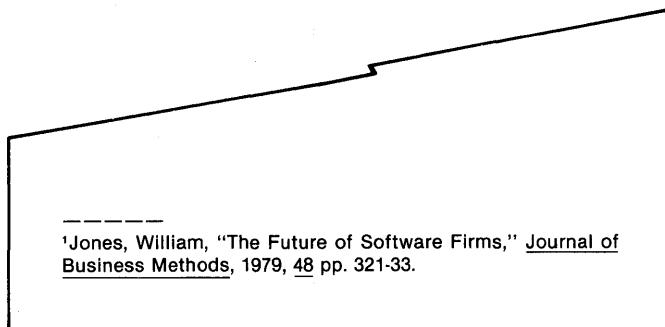
---

#### Command Form:

....As Jones has observed, software firms have an enormous potential, as yet largely unrealized. @FOOT[Jones, William, "The Future of Software Firms," @U(Journal of Business Methods), 1979, @U(48) pp. 321-33.]...

---

#### The Result:



## **@NOTE**

### **The NOTE Command**

Causes text to be printed as a numbered note at the end of the document. A reference number to the note is inserted in the text at the location of the command.

The 'footpush' style parameter (see Chapter XVIII, page 3) determines whether this number is printed as a superscript (default) or is enclosed in brackets, [ ].

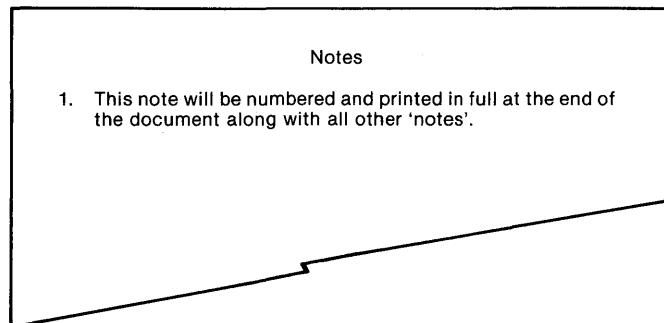
---

#### **Command Form:**

.....@NOTE(This note will be numbered and printed in  
full  
at the end of the document along with all other 'notes.')  
....  
.....

---

#### **The Result:**



## PAGEHEADINGS/PAGEFOOTINGS

### **@PAGEHEADING @PAGEFOOTING**

In long documents it is often desirable to print 'pageheadings' and 'pagefootings' at the top and bottom of pages, in order to identify not only the work as a whole, but also the chapters and sections that are included.

Perfect Writer provides a simple and convenient means of doing this through its PAGEHEADING and PAGEFOOTING format commands. Each command has three parts:

- a left justified part
- a center part
- a right justified part

The words "left=", "center=", and "right=" are used to specify the text to be placed in each part. If any of these words are omitted, the particular space in the heading or footing that it designates will be left blank. In addition, a "line=" designation may be added to include a line of text beneath a page heading. 'Center,' 'right,' and 'line' designations may **not** include tab or newline characters.

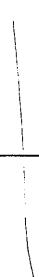
**When specifying headings and footings for an entire document, the commands are placed at the beginning of the document.**

### **Page Numbering**

If no PAGEFOOTING command is given, as in the above example, Perfect Writer will automatically number all pages sequentially, beginning with '-1-'. If you wish to turn off the automatic page numbering, then simply place an empty PAGEFOOTING at the beginning of a document. This would appear as:

**@PAGEFOOTING( )**

---



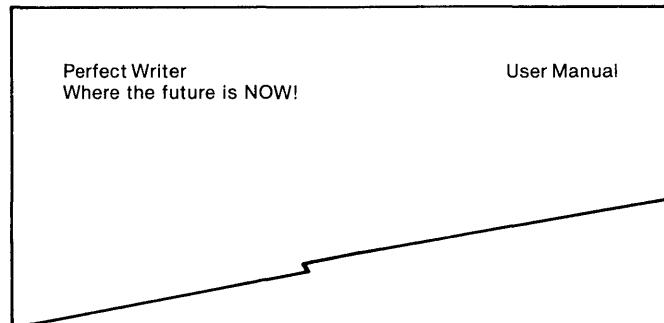
**Command Form:**

@PAGEHEADING(left = "Perfect Writer", right = "User Manual", line = "Where the future is NOW!")

---

**Result:**

This would result in the following pageheading appearing on every page in the document.



## Variable Pageheadings and Pagefootings

It is possible to have pageheadings and pagefootings vary throughout the document, depending upon the material being presented in any particular chapter or section.

### VARIABLE PAGEHEADING

The following PAGEHEADING command, placed at the beginning of the document, would cause the current chapter title to be printed in the center of every page of that chapter:

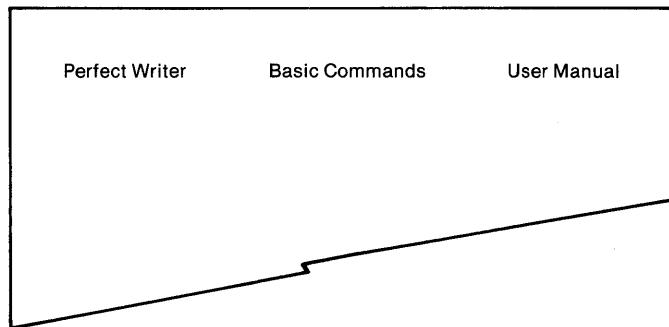
---

#### Command Form:

```
@PAGEHEADING(left = "Perfect Writer",
center = @VALUE{CHAPERTITLE}, right = "User Manual")
```

---

#### Result:



Here the predefined variable name CHAPERTITLE (see Chapter XVII, page 10) is used in conjunction with the @VALUE command (see Chapter XVII, pages 11-12) to insert the current chapter title in the center space of the heading. As the chapters change, so will the chapter title being printed in the heading.

---

## **VARIABLE PAGEFOOTING**

The following PAGEFOOTING command, placed at the beginning of a document, would cause each chapter of the document to be numbered separately.

---

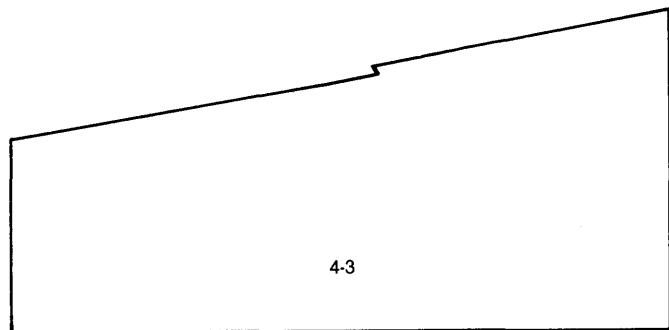
### **Command Form:**

---

```
@PAGEFOOTING[center = “@REF(CHAPTER)-@VALUE(page)”]
```

---

### **Result:**



Here, the value of two predefined 'counters' are used—"chapter" and "page." (For a discussion of counters, see Chapter XVII, page 16.) As the chapters and pages change, the values of these two counters will change also.

---

## Odd-Even Pageheadings

Pageheadings can be different for **odd**- or **even**-numbered pages of a document. The words "odd" and "even" are appended to the PAGEHEADING and PAGEFOOTING commands to specify which page the heading will be for. This option is often used when pages are duplicated on both sides of a sheet and gathered into a book.

---

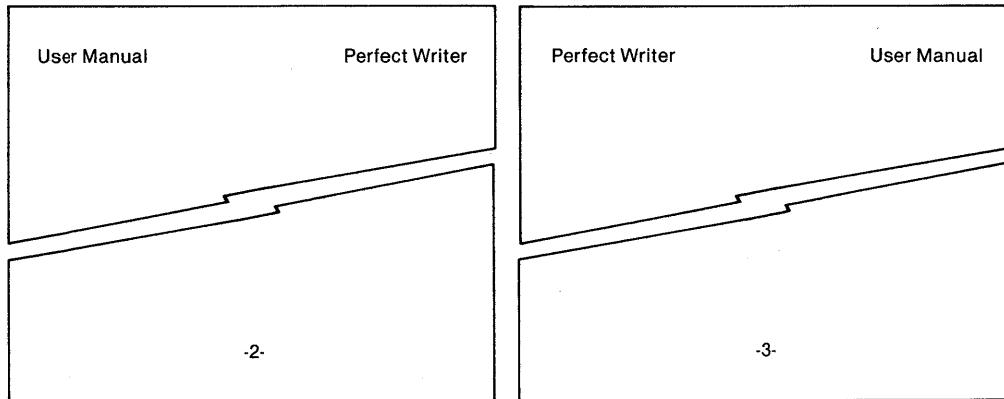
### Command Form:

`@PAGEHEADING(left = "Perfect Writer", right = "User Manual" {odd})`

`@PAGEHEADING(left = "User Manual", right = "Perfect Writer" {even})`

---

### Result:



## Chapter XVII

### TOOLS FOR FORM LETTER DESIGN

Perfect Writer provides several commands and features which are useful in the construction of form letters. In this chapter we will introduce these along with Perfect Writer's use of variables in formatting.

#### **@CASE**

##### **The @CASE Command**

The CASE command procedure allows one of several alternate portions of text in a document to be selected for printing. Thus, a document can have several variations of text 'built-in,' and each one can be selected as desired.

The CASE formatting option is useful in the preparation of form letters or multi-purpose documents where individual variations are necessary to distinguish separate versions of what is essentially the same document.

##### **Example:**

Suppose that a letter has been prepared to introduce Perfect Writer to a variety of customers: secretaries, businessmen, college professors, students, and the 'general user' (this last can include all of the previous categories).

Overall the letter does not distinguish between the categories of user, but simply presents the advantages Perfect Writer brings to the task of writing and editing.

It is desirable, however, that the final paragraph address itself directly to only one potential user, thus personalizing what would otherwise be an impersonal form letter.

---

**Steps:**

1. Create at the end of the document the several versions of the final paragraph, tagging each with a name 'tag,' and enclosing all of them within the 'fences' of a CASE command. Thus:

@CASE(User,

Secretary, "Perfect Writer will do more than just eliminate clerical drudgeries. It will enhance your abilities in all areas, making you not only more capable and valuable professionally, but as well allow you more creative control in editing and designing documents."

Businessman, "As you can see, it is no exaggeration when we say that Perfect Writer will increase the efficiency of your office and business routines in ways that you never dreamed possible."

Professor, "As familiar as you are with drafting and preparing written documents, we feel that you will be particularly appreciative of these many time-saving innovations that Perfect Writer brings to the task and of writing and editing. Perfect Writer is particularly well-suited to the task of preparing book length manuscripts and professional articles."

Student, "What Perfect Writer means to you is faster and more efficient preparation of the notes, papers, and examinations that your studies require. At last, outstanding academic achievement is no longer tied to one's endurance at the typewriter."

NULL, "[ ]".

Else, "From all of this it is easy to see that Perfect Writer fulfills most, if not all, of the needs of individuals who depend upon fast, efficient written communications to accomplish their private and professional goals.")

---

2. Select the desired text by assigning **at the beginning of your document** a 'value' to the variable name "User," using a @STRING command (see Chapter XVII, pages 11-13). For example:

1. **@STRING(User = 'Professor')**

The 'professor' text will be selected and printed for the final paragraph.

2. **@STRING(User = 'Student')**

The 'student' text will be selected and printed for the final paragraph.

3. **@STRING(User = NULL)** No final paragraph will be printed.

4. **@STRING(User = )** [i.e. That is, you neglect to assign a value to the variable name 'User.']}

The 'Else' option will be selected and printed.

In this example notice the following:

- A single pair of fences enclose all of the alternative texts following the CASE command.
- Each alternative text is enclosed in quotation marks, "", and is ended with a period. It is also separated from its name 'tag' by a comma.
- A 'NULL' option has been included to allow for the case when you may not want to include a final paragraph at all. (The NULL case is specified in the CASE command by two facing brackets, [ ], containing nothing, not even spaces.)
- The 'Else' option contains the text for a general user, that is, when no other particular user is specified.
- The word 'User' which follows immediately after the @CASE command is the variable name, that, when given a value, will select the desired text.

## **@INCLUDE**

### **The @INCLUDE Command**

The @INCLUDE command instructs Perfect Writer to insert a document from another file into the text of the document file being printed. The insertion is made at the point in the text where the command is given. Following the insertion of the file defined in the @INCLUDE command, Perfect Writer continues printing the current document.

The @INCLUDE command can also be used to assemble a manuscript length document stored in several different files (perhaps a different file for each chapter). With a master file made of general format commands, style parameter settings, and @INCLUDE commands for each different file, it would be possible to control the assembling of the full document. Documents should generally be 20 pages or less. Documents longer than 20 pages should be divided into parts and assembled during printing with the @INCLUDE command.

---

#### **Command Form:**

The INCLUDE command can be given singly or in combination. For example,

---

**@INCLUDE(filename.MSS)**

---

The INCLUDE command can also be used in conjunction with the MESSAGE command to accept additional text typed at the console during printing. (See discussion below.) For example,

**@INCLUDE(con:)**

would instruct Perfect Writer to include text taken interactively from the CONsole while the text was being formatted, rather than to take the text from a file stored on disk.

---

## **@MESSAGE**

### **The MESSAGE Command**

Causes Perfect Writer to display on the console screen, during printing, messages that query for additional information. Such messages are used in conjunction with the INCLUDE command. This command procedure is most often used in personalizing form letters and other standardized documents.

---

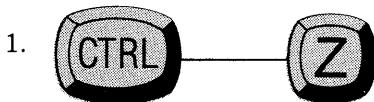
#### **Command Form:**

```
@MESSAGE(What is the address?)  
@INCLUDE(CON:)
```

---

When, during printing, this command is encountered, Perfect Writer will display the message, "What is the address?" on the console screen. At that time you should enter the required information. The @INCLUDE(CON:) instructs Perfect Writer to insert into the text being printed whatever information you type at the console ("CON").

**General Note:** Since Perfect Writer's regular editing commands are inoperative during printing, the following special commands should be used when inputting text at the console in response to an @INCLUDE(CON:) message:



Instructs Perfect Writer that the required information has been entered and printing can resume.

---

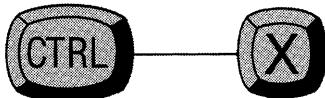
2.



'carriage return'

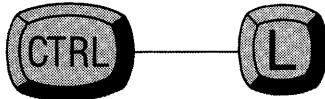
Tells Perfect Writer to begin a new line for the information you are inputting.

3.



Deletes the characters of the last line entered but not the line itself.

4.



Reprints all text entered.

## **@COMMENT**

### **The COMMENT Command**

Causes Perfect Writer to ignore a specified portion of text. This command is useful for providing internal notes to the author of the document. It can also be used to temporarily omit portions of a document from printing, without the necessity of having to delete them.

---

#### **Command Form:**

```
.....  
.....  
@COMMENT(Don't forget to add details about...  
whatever it is!!!)  
.....  
.....
```

---

**Note:** If the omitted text contains other embedded format commands, use the BEGIN/END command form, since other format 'fences' may disrupt the execution of this command.

---

## @BLANKPAGE

### The BLANKPAGE Command

Causes Perfect Writer to insert additional page(s) after the current page is finished. Additional blankpages added with this command will have pageheadings and pagefootings and be numbered. To get completely blank pages, do not use this command, but simply insert blank sheets manually after the document is printed.

---

#### Command Form:

**@BLANKPAGE(4)**

---

#### Result:

This will insert four numbered but blank pages.

## **@BLANKSPACE**

### **The BLANKSPACE Command**

Causes Perfect Writer to insert a specified number of blank lines into the text where the command is given.

---

#### **Command Form:**

**@BLANKSPACE(3 lines)**

---

#### **Result:**

Three blank lines would be inserted into the text at the place where the command is given.

## @NEWPAGE

### The NEWPAGE Command

Causes Perfect Writer to print the next line after this command at the top of a new page. this command is useful in selecting when to switch to a new page.

---

#### Command Form:

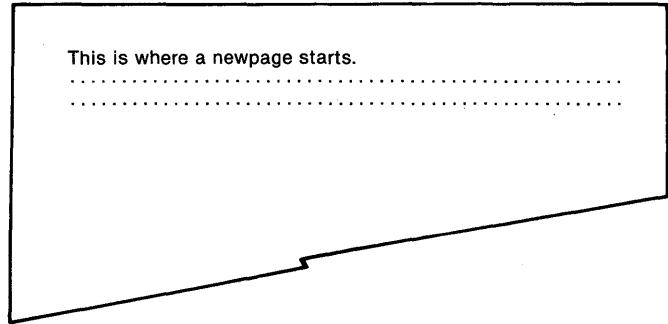
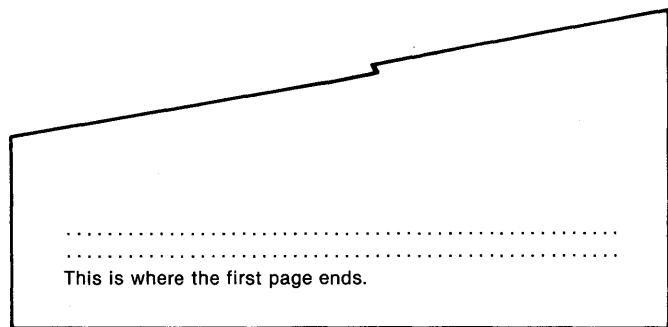
This is where the first page ends.

@NEWPAGE

This is where a newpage starts.

---

#### Result:



## **USING VARIABLES IN FORMATTING**

Perfect Writer offers a unique formatting option that allows the use of 'alphanumeric variables,'—that is, strings of numbers or words that, for one reason or another, change during the course of a document. The following examples illustrate the use of 'variables.'

### **The STRING/VALUE Formatting Procedure (for use with Word variables)**

Suppose that you are preparing a form letter that will be sent to a number of similar institutions. In each version of the letter the name of the institution will be mentioned perhaps several times.

Through a simple command procedure, you can instruct Perfect Writer to insert the name of each separate institution into each of the different letters as it is prepared.

Two commands are used to accomplish this:

1. **@STRING:** At the beginning of the document, it assigns a name to a longer string of characters or words. For example,

`@STRING(School = "Massachusetts Institute of Technology")`

This STRING command tells Perfect Writer that throughout the first document being printed the variable name 'school' will stand for "Massachusetts Institute of Technology."

2. @VALUE: Causes the string of words assigned to the variable name by the @STRING command to be inserted into the text. For example,

---

**Command Form:**

There are several major centers for research on artificial intelligence. Among these @VALUE(school) is considered the leader. Much of the fundamental research derives from work done at @VALUE(school).

---

**Result:**

There are several major centers for research on artificial intelligence. Among these Massachusetts Institute of Technology is considered the leader. Much of the fundamental research derives from work done at Massachusetts Institute of Technology.

---

The great convenience in using this STRING/VALUE procedure is that when the time comes to change the institution name, there is no need for a search and replace procedure, followed by a retyping. Simply reassign a new institution name to the variable 'school,' using the same @STRING format command at the beginning of the document.

`@STRING(school = Stanford University)`

Perfect Writer automatically makes the substitution everywhere in the letter.

---

**Command Form:**

There are several major centers for research on artificial intelligence. Among these @VALUE(school) is considered the leader. Much of the fundamental research derives from work done at @VALUE(school).

---

**Result:**

There are several major centers for research on artificial intelligence. Among these Stanford University is considered the leader. Much of the fundamental research derives from work done at Stanford University.

---

## Predefined Word Variables

Perfect Writer 'predefines' certain word variables, which means that a @STRING command is not needed to assign values, since they automatically take their values from the context of the document you are writing. These predefined word variables are:

CHAPERTITLE	The title of the current chapter.
SECTIONTITLE	The title of the current section.
SUBSECTIONTITLE	The title of the current subsection.
PARAGRAPHTITLE	The title of the current paragraph.
APPENDIXTITLE	The title of the current appendix.
APPENDIXSECTIONTITLE	The title of the current appendix section title.

When used with the @VALUE command, these predefined word variables insert the title of the current chapter, section, paragraph, etc., into the text. They are particularly useful with the PAGEHEADING and PAGEFOOTING format commands (see Chapter XVI, pages 12-16 for examples of these commands).

## @TITLE The TITLE Command

Causes the name of the current section to be inserted into the text. Most useful as a 'dummy' in specifying a title in a pageheading or a pagefooting when that title has not yet been selected, or is likely to be changed in the course of editing. Uses ONLY the following sectioning names: CHAPTER, SECTION, SUBSECTION, PARAGRAPH, APPENDIX, and APPENDIXSECTION.

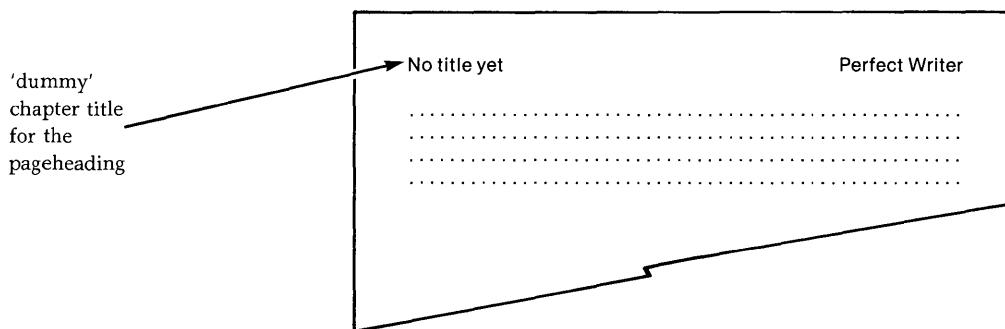
---

### Command Form:

```
@CHAPTER(No title yet)  
@PAGEHEADING(Left = "@TITLE(CHAPTER)", Right =  
"Perfect Writer")
```

---

### Result:



## The SET/REF Formatting Procedure (for use with Numeric variables)

Numeric variables find their greatest usefulness in long documents, where 'backward references' need to be made—that is, references to portions of text in earlier chapters, sections, paragraphs, etc.

Perfect Writer will automatically determine the page number (chapter, section or paragraph number) where an earlier reference occurred, and insert it at the specified point in the document.

To do this Perfect Writer makes use of 'counters' which automatically keep track of chapter numbers, page numbers, section numbers, etc. These 'counters' are appropriately named:

page  
chapter  
section  
subsection  
paragraph  
appendix  
appendixsection

Two commands are used in connection with counters to achieve backward referencing:

1. **@SET:** This command assigns a variable name of your choice to a counter.
2. **@REF:** This format command accomplishes the backward reference by inserting the value of the counter assigned by @SET into the forward position in your document.

Though this formal description sounds involved, the procedure is actually quite simple.

**Example:**

Suppose that early in your document you mention some item—a name, concept, idea, whatever—that you know you will later be referencing. In that later reference you will want to cite the page, chapter, or section number on which your mention of the item occurred.

Normally this would present a difficulty, since your document has not yet been typed, and therefore not apportioned into pages, sections, etc. Until now the solution has been to fill in the backward page references as you type the document. But this is risky, because suppose you later add new material which changes the paging. You must go through the entire procedure again of finding and correcting all of your backward page references.

Perfect Writer takes care of this problem automatically in the following easy steps:

1. Using the @SET command, tag the first mention of the item to be referenced with a variable name, setting this variable name equal to the appropriate 'counter' you wish to reference by (page, chapter, section, etc.). For example,

There are several major centers for research on artificial intelligence. Among these, Massachusetts Institute of Technology is considered the leader. Much of the fundamental research derives from work done at Massachusetts Institute of Technology. @SET(MIT = page)

Here you have tagged a discussion of the 'Massachusetts Institute of Technology' with the variable name "MIT," and have set this variable name equal to the counter 'page.'

2. Later in your document when you wish to refer to the page on which this discussion of MIT occurred, insert the @REF command in the following way:

---

**Command Form:**

..... as has been discussed  
(see page @REF[MIT]), the Massachusetts Institute of Tech-  
nology represents .....

---

**Result:**

..... as has been discussed  
(see page 17), the Massachusetts Institute of Technology  
represents .....

---

The SET/REF format procedure is not affected by the addition or subtraction of pages. Because only variable names have been used, changes in the overall document pagination are automatically adjusted for. Thus, once having placed them you never again have to worry about the correctness of your backward reference.

The SET/REF format commands (with counters) are particularly helpful when formatting pageheadings and pagefootings.

---

## Chapter XVIII

### **DOCUMENT STYLE COMMANDS**

Document style commands control the printed output of your document, including such aspects as margin settings, footnote style, line spacing, justification, indentation, paper size, the printing device to be used, and more.

#### **The DEVICE Command**

Causes Perfect Writer to format your document to the specifications of the named printing device that was described during installation of Perfect Writer. (See Appendix A, page 7, for a discussion of 'devices'.)

If used, the DEVICE command must be the **first** command, preceding all other text of a document.

---

#### **Command Form:**

**@DEVICE(DeviceName)**

---

The device named must have been defined with the PFCONFIG.COM program. Perfect Writer will format the document for the named device (see Installation Appendix.). You may configure more than one device.

## STYLE PARAMETER Commands

Style parameter commands allow you to alter Perfect Writer's formatting default settings to your particular tastes, by changing such things as margins, indentions, line spacing, and spacing between paragraphs, not only for the document as a whole, but for each of the format options that may have been used in the document.

Style parameter commands, if used, can appear **ONLY AT THE BEGINNING** of your document, **following the @DEVICE command**, and preceding everything else.

---

### Command Form:

Style parameter commands can be given singly or in combination. For example:

`@STYLE(bottommargin 3 lines)`

`@STYLE(topmargin 3 lines)`

`@STYLE(leftmargin 8 chars)`

or

`@STYLE(bottommargin 3 lines, topmargin 3 lines, leftmargin 8 chars, . . . etc.)`

---

Notice the spacing that style parameters take. **Words are merged** (i.e. bottommargin, **not** bottom margin). When used in combination each style parameter is separated from the others by a comma.

Many style parameters use 'dimensioned numbers' to express the values they hold, such as: "3 lines," "8 characters" (or "8 chars"). Some style parameter commands present 'Yes/No' options, while others require a word or words to define their value.

Following are the various style parameter commands. Any of these can be altered as you desire, using the command form shown. The values given in the examples are the default values that are automatically provided at the time of configuration.

---

Above	The number of lines that precede certain format options, in particular: General text formats and lists.
	@STYLE(above 1 line) [default]
Below	The number of lines that follow certain format options, in particular: Environment and list formats.
	@STYLE(below 1 line) [default]
Bottommargin	The distance between the last text printed on the page (including pagefootings) and the bottom of the sheet of paper.
	@STYLE(bottommargin 3 lines) [default]
Chapter (or Chapters)	Determines whether section numbering of a document begins with chapters and appendices, or with sections and appendixsections. If "yes," chapters and appendices are numbered. If "no," numbering begins with sections and appendixsections.
	@STYLE(chapters yes) [default]
Footerspacing	The distance between the end of the running text and the pagefooting.
	@STYLE(footerspacing 3 lines) [default]
Footpush	Determines how note and footnote reference numbers are placed in the text. If "yes," a superscripted number is used. If "no," the number is enclosed in brackets, [ ].
	@STYLE(footpush yes) [default]

Headerspacing	The distance between the pageheading and the beginning of the running text.
	@STYLE(headerspacing 3 lines) [default]
Indent or Indentation	Determines how far the first line of a paragraph is indented in TEXT and QUOTATION formats, as well as how far the first line is <b>undented</b> from the paragraph in the UNDENT format.
	@STYLE(indent 2 chars) [default]
Justification	Determines whether the right margin will be justified in formats that 'fill' the lines. If "yes," the right margin will be made even and straight. If "no," the right margin will be left ragged.
	@STYLE(justification yes) [default]
Leftmargin	The distance between the left edge of the sheet of paper and the beginning of any printing.
	@STYLE(leftmargin 8 chars) [default]
Levelhang	Specifies the form of paragraphs in the LEVEL format. If "yes," the paragraph number is undented from the body of the paragraph. If "no," the number is included in the body of the paragraph.
	@STYLE(levelhang no) [default]
Levelindent	Specifies whether the paragraphs of nested LEVEL formats should be indented from each other. If "yes," nested format paragraphs are indented. If "no," they are not.
	@STYLE(levelindent yes) [default]

Linewidth	Specifies the length of a printed line. This distance is added to the leftmargin to obtain a new rightmargin. If later the rightmargin or paperwidth parameter is given, the value given here will be discarded.
	<b>@STYLE(linewidth __ characters)</b> [No default]
Notes	Specifies where footnotes are to be placed. If "bottom," they are placed at the bottom of the page where the reference occurs. If "endnote," they are placed together at the end of the printed document. If "inline," they are placed in the text at the point of reference, enclosed in brackets, [ ].
	<b>@STYLE(Notes bottom)</b> [default]
Paperlength	The height of the sheet of paper being used.
	<b>@STYLE(paperlength 11 inches)</b> [default]
Paperwidth	The width of the sheet of paper being used.
	<b>@STYLE(paperwidth 9 inches)</b> [default]
Rightmargin	The distance between the last printed character and the right edge of the paper.
	<b>@STYLE(rightmargin 8 chars)</b> [default]
Scriptpush	Determines whether extra lines are left above or below lines containing super- and subscripts. If "yes," an extra blank line will be inserted (to help prevent confusing a superscript on one line with a subscript on the previous line, and vice versa). If "no," the extra line will not be inserted. This command is automatically set to "no" for devices that cannot half-space vertically.
	<b>@STYLE(scriptpush yes)</b> [default]

Spacing      Determines the spacing between lines. Using "2 lines" causes double spacing, "3 lines" triple spacing, etc. Do not set this value to less than one line.

**@STYLE(spacing 1 line) [default]**

Spread      Determines the distance (in lines) between paragraphs.

**@STYLE(spread 1 line) [default]**

Topmargin      The distance between the top of the sheet of paper and the first line of text, including the pageheading if it is used.

**@STYLE(topmargin 3 lines) [default]**

---

## Chapter XIX

### **DOCUMENT DESIGN LESSONS**

---

This set of lessons is designed to introduce you to Perfect Writer's document design system. The easiest way to learn a new program is by using it, so these lessons will primarily consist of source documents (the file you create on disk) and formatted results.

Each lesson after Lesson 1 is an example. Examples show a source document and the output you get after running it through Perfect Writer. At the bottom of the example source document page is a description of each new command or feature used. The example source documents are stored on the Perfect Writer master disk, so don't be afraid to play around with the copies on your personal disk—take each new example and change it around, to see what you get as output.

We think that after becoming familiar with the document design features of Perfect Writer and using them on one or two documents of your own, you will use them as a matter of course, every time you compose text on the computer.

## LESSON 1

In this lesson, you will learn how to run Perfect Writer to produce output on your terminal.

Input document files for Perfect Writer always have the extension ".MSS." This stands for "ManuScript Source." For example, the file we will be working with in this lesson is called "TEXT1.MSS." Remember that it may be called "B:TEXT1.MSS" or the like, if you have a multiple-disk computer system.

In order to print a document with Perfect Writer you must select option F, "format a file" from the Main Selection Menu. Perfect Writer will respond by asking for the filename of the document to be printed. After you enter the filename, Perfect Writer will transfer you to the Perfect Format Selection Menu. From this menu you may elect to print a copy to either the console (option C) or the printer (option P). When you print directly from the format program, Perfect Writer will not be able to use all of the features available on your printer. For our purposes here, however, you will get an adequate copy of what is in your file.

Try printing a document now. From the main menu select F, to format a file, and then enter the filename "TEXT1.MSS". Since Perfect Format assumes the ".MSS" extension, all you have to enter is "TEXT1". From the Perfect Format menu select the console option ("C") and then type "G" for the program to go. The output from Perfect Format should look like the formatted document in Figure 2. If all goes well, then print Lesson 1 on your printer with the P option.

You should note that unless you select the VERBATIM option (which prints text exactly as it appears on the screen) from the Perfect Formatter Selection Menu, Perfect Writer will group text into justified paragraphs. Text that you do not want in justified paragraphs needs to be formatted using one of the other formats such as VERBATIM, ADDRESS, FLUSHLEFT, EXAMPLE, DISPLAY, QUOTATION and the other environments described in the lessons to follow.

---

Several error messages are possible, though unlikely, at this point. They are:

- Can't open file TEXT1.MSS. This may occur if the disk you are using has no lesson input text on it. Take a look at the directory for your disks (use the D option) and make sure that you have typed the filename correctly.
- Can't find the configuration file. If this happens, find the person who installed the Perfect Writer program and ask him or her to help you. The Installation Guide may be of some help here.

Notice that the first page shows you the input to Perfect Writer, and the second the output from it. You can examine the input by selecting option E from the main menu and entering the TEXT1.MSS filename.

Notice that the first page shows you the input to Perfect Writer, and the second the output from it. You can examine the input by selecting option E from the main menu and entering the TEXT1.MSS filename.

### The Input File:

@VERBATIM {The improvement in human knowledge and survival potential following the invention of writing was immense. There was also an improvement in self-reliance: It is possible to learn at least the basics of an art or a science from a book and not be dependent on the lucky accident that there is a nearby master craftsman to whom we may apprentice ourselves.}

When all is said and done, the invention of writing must be viewed not only as a brilliant innovation but as a surpasssing good for humanity. I believe the same will be said of those who are today devising computers and programs at the edge of machine intelligence. The next major structural development in human intelligence is likely to be a partnership between intelligent humans and intelligent machines (from The Dragons of Eden by Carl Sagan, pages 224-5).

---

TEXT1.MSS

## The Output:

The improvement in human knowledge and survival potential following the invention of writing was immense. There was also an improvement in self-reliance: It is possible to learn at least the basics of an art or science from a book and not be dependent on the lucky accident that there is a nearby master craftsman to whom we may apprentice ourselves.

When all is said and done, the invention of writing must be viewed not only as a brilliant innovation but as a surpassing good for humanity. I believe the same will be said of those who are today devising computers and programs at the edge of machine intelligence. The next major structural development in human intelligence is likely to be a partnership between intelligent humans and intelligent machines (from The Dragons of Eden by Carl Sagan, pages 224-5).

---

### TEXT1

#### **Important**

Note that the paragraphs have been either "filled" or "justified."

- "Filling" text means reading words from the manuscript file and putting as many as will fit on a single document line. This process will make each line as long as possible within the margins, just as a good typist would. The body of this definition is an example of filled text.
  - "Justifying" text means taking a filled line and inserting extra space between words (or letters) in order to have a right margin perfectly aligned, somewhat akin to a newspaper column of print. The body of this definition is an example of justified text. This is the standard default. If you do not want this, then select the VERBATIM option from the 'Format Selection Menu.'
  - Note that paragraph indentation is always uniform in the output, regardless of how the input looks. Likewise the blankspace in between paragraphs is always the same, regardless of how the input document looks.
  - The choices about filling, justifying, spacing, and indentation have been set up for you in what is called the "configuration file." These can be changed. See the Style commands.
-

the document's purpose. The first step in creating a document is to determine its purpose. This is the most important decision you will make about your document. It is also the most difficult decision to make because it requires you to think about the needs of your audience. You must consider what information they need to know and how they will use it. You must also consider what message you want to convey and how you can best do that. Once you have determined the purpose of your document, you can begin to plan the layout and design.

The second step in creating a document is to plan the layout and design. This involves deciding on the overall structure of the document, including the title page, table of contents, headings, and body text. It also involves deciding on the font, size, and style of the text, as well as the placement of figures and tables. The layout and design should be consistent throughout the document, and should reflect the purpose of the document.

The third step in creating a document is to write the content. This involves gathering all the necessary information and organizing it into a logical flow. The content should be clear and concise, and should be presented in a way that is easy to understand. The content should also be relevant to the purpose of the document.

The fourth step in creating a document is to proofread and edit the content. This involves checking for errors in spelling, grammar, and punctuation. It also involves checking for consistency in the layout and design. The content should be reviewed by someone else to ensure that it is accurate and complete.

The fifth step in creating a document is to print or publish the document. This involves choosing the right format for the document, such as a printed book or an electronic document. It also involves choosing the right printing or publishing service. The document should be printed or published in a way that reflects the purpose of the document.

The sixth step in creating a document is to evaluate the document. This involves reading the document and determining if it meets the purpose of the document. If it does not, then the process should be repeated until it does. The evaluation should be done by someone who is familiar with the purpose of the document.

The seventh step in creating a document is to make changes based on the evaluation. This involves making changes to the layout, design, or content of the document to reflect the findings of the evaluation. The changes should be made in a way that reflects the purpose of the document.

The eighth step in creating a document is to repeat the process. This involves repeating the steps of planning, writing, proofreading, and evaluating until the document meets the purpose of the document. The process should be repeated until the document is complete and meets the needs of the audience.

---

## **LESSON 2: A Letter**

This lesson uses several simple Perfect Writer "format commands." Format commands always begin with the at-sign, "@." They may appear anywhere in the text. To Perfect Writer's document design program, anything that begins with an @-sign is a command; therefore, in order to insert a real @-sign into your document, you must type a double-@ ("@@") in your manuscript file.

Perfect Writer uses two types of format commands:

- environment**      The environment format command affects the form of the output document for a particular block of input manuscript text. This type of format command is called an environment command. The text affected by the format command is fenced, with the text between the fences becoming the environment formatted by the command.
- directive**      The directive format command issues a specific directive to change the output document, and is not related to any particular fenced portion of text in the input manuscript.

In order to see the result of formatting a particular letter, select F from the main menu and enter the filename:

TEXT2.MSS

Select option P and then G to start the printing of Lesson 2.

In addition, as you look at the console during printing, you will see numbers while the text is being formatted. These reflect the page the formatting is on at the time.

### The Input File:

@ADDRESS (123 Main Street<sup>1</sup>  
New York, NY 10007)

@FLUSHLEFT(Robert Hemingway<sup>2</sup>  
1234 Main Street  
Centerville, IL 60601)  
@FLUSHLEFT(Dear Mr. Hemingway;  
@TEXT{ Thank you for your recent letter and comments about  
Perfect Writer. We are delighted to  
receive such comments and, as you can tell, we too are pleased  
with the performance of Perfect Writer.

Your suggestions and comments are very useful. Thanks  
again for your consideration. }<sup>3</sup>  
@CENTER(Be sure to see our other software!)  
@CLOSING[Cordially,

Don Cochran ]

---

TEXT2.MSS

<sup>1</sup> The ADDRESS environment is followed immediately by the "fences" surrounding the text. (No spaces are allowed between the command name and the fence.) In this case, the fences are "[" and "]". As you can see in the CLOSING environment at the bottom, "[" and "]" work as well. We used them there because of the parentheses inside the text. Likewise, single and double quotes can be used as text fences for formatting environments. In general, use a different fence when it appears within the text being formatted.

<sup>2</sup> The FLUSHLEFT environment command places each line against the left margin. There is also a FLUSHRIGHT environment. CLOSING, ADDRESS, and CENTER all place the text in appropriated places on the page—see the output.

<sup>3</sup> Any text not enclosed in any other environment (here, the main body of the letter) is formatted the same way as what you saw in Lesson 1 (i.e., it is grouped into justified paragraphs).

---

**The Output:**

123 Main Street  
New York, NY 10007

Robert Hemingway  
1234 Main Street  
Centerville, IL 60601

Dear Mr. Hemingway:

Thank you for your recent letter and comments about Perfect Writer. We are delighted to receive such comments and, as you can tell, we too are pleased with the performance of Perfect Writer.

Your suggestions and comments are very useful. Thanks again for your consideration.

Be sure to see our other software!

Cordially,

Don Cochran

---

TEXT2

---



## **Lesson 3**

### **BOILERPLATE LETTER AND FORM LETTER**

This lesson illustrates the preparation of documents that use pieces of other documents and console input to produce the finished product.

Two new commands are used in the examples which follow: "@INCLUDE" and "@MESSAGE." The INCLUDE command allows text to be taken from another .MSS file and inserted into the current file at that point. The MESSAGE command prints out a message to the console.

The first example shows a letter made up of sections of standard boilerplate text. (The word "boilerplate" comes from journalism, where standard typeset features were usually available premade in plates or mats, rather than set by hand.) To look at the output, select option F and enter the filename, "TEXT3A.MSS" from the format menu, select options P and then G.

### The Input File:

```
@INCLUDE(ADDRESS.MSS)1
@FLUSHLEFT(Robert Hemingway
1234 Main Street
Centerville, IL 60601)
```

```
@FLUSHLEFT(Dear Mr. Hemingway:)
```

Thank you for your recent letter and comments about  
Perfect Writer. We are delighted to receive such comments, and  
as you can tell, we too are pleased with the software engineering  
advances embodied in Perfect Writer.

You mentioned an interest in Perfect Speller and our  
family of related software. Let me briefly tell you about each.

```
@INCLUDE(SPELLER.MSS)
@INCLUDE(FILER.MSS)
@CENTER(Again, we appreciate your kind words.)
@CLOSING[Sincerely,
```

Don Cochran]

---

TEXT3A.MSS

---

<sup>1</sup> If you use Perfect Writer to edit the files ADDRESS.MSS, SPELLER.MSS, and FILER.MSS, you will see the  
text which was originally included in our letter.

## The Output:

123 Main Street  
New York, NY 10007

Robert Hemingway  
1234 Main Street  
Centerville, IL 60601

Dear Mr. Hemingway:

Thank you for your recent letter and comments about Perfect Writer. We are delighted to receive such comments, and as you can tell, we too are pleased with the software engineering advances embodied in Perfect Writer.

You mentioned an interest in Perfect Speller and our family of related software. Let me briefly tell you about each.

### PERFECT SPELLER

Perfect Speller is an automatic proofreader which checks your editing against an equivalent 50,000 word dictionary. After editing a text in Perfect Writer a simple keystroke command and the text is checked against Perfect Speller's dictionary for mismatches. It's as simple as that.

### PERFECT FILER

Perfect Filer is a full feature mailing list and information management system. Form letters and documents prepared with Perfect Writer can be managed and prepared by Perfect Filer.

Again, we appreciate your kind words.

Sincerely,

Don Cochran

---

TEXT3A

---

### Form Letter with Console Input

For the next example, you will be asked to enter at the console specific information needed for the letter being formatted. Type in the requested information from the console. As you enter information you may make mistakes. In this case, a few keys have special meanings (see pages XVII-5 to XVII-6):

- When you type a Control-Z, this tells Perfect Writer that you are done entering console input.
- If you want to erase the last character you typed, type the backspace or delete key.
- If you want to erase the entire line you have just been typing, type a Control-U. (In order to get back to the previous line of typing, you can hit the delete or backspace keys to get rid of the carriage return.)
- If you want to see everything you have just typed in order to proofread it just before typing the Control-Z, type Control-R (for Redisplay) and Perfect Writer will play back your typing up to that point.

To get started, select F and enter TEXT3B.MSS. Then select options P and then G.

#### The Input File:

```
@INCLUDE(ADDRESS)
@FLUSHLEFT[
@MESSAGE(Enter full name and address:

@INCLUDE(CON:)']
@FLUSHLEFT[Dear @MESSAGE(Enter first name):@INCLUDE(CON:)]
    Thank you for your recent letter and comments about
Perfect Writer. We are delighted to receive such comments and,
as you can tell, we too are pleased with the software engineering
advances embodied in Perfect Writer.
```

You mention an interest in @MESSAGE(What software is  
client interested in?)@INCLUDE(CON:) and our family of related  
software. Let me briefly tell you about each.

```
@INCLUDE(SPELLER)
@INCLUDE(FILER)
@CENTER(Be sure to see our software at your local computer store!)
@CLOSING[Cordially,
Don Cochran]
```

---

TEXT3B.MSS

<sup>1</sup> The INCLUDE to "CON:" means to include text taken from the CONsole rather than a text file.

## The Output:

123 Main Street  
New York, NY 10007

Robert Hemingway  
1234 Main Street  
Centerville, IL 60601

Dear Bob:

Thank you for your recent letter and comments about Perfect Writer. We are delighted to receive such comments, and, as you can tell, we too are pleased with the software engineering advances embodied in Perfect Writer.

You mentioned an interest in Perfect Speller and our family of related software. Let me briefly tell you about each.

### PERFECT SPELLER

Perfect Speller is an automatic proofreader which checks your editing against an equivalent 50,000 word dictionary. After editing a text in Perfect Writer a simple keystroke command and the text is checked against Perfect Speller's dictionary for mismatches. It's as simple as that.

### PERFECT FILER

Perfect Filer is a full feature mailing list and information management system. Form letters and documents prepared with Perfect Writer can be managed and prepared by Perfect Filer.

Be sure to see our software at your local computer store!

Cordially,

Don Cochran

---

TEXT3B

---



## LESSON 4: An Office Memorandum

This lesson will introduce several commands for making lists or descriptions of things.

### The Input File:

```
@STRING(SUM = "Perfect Writer")  
 @BEGIN(FLUSHLEFT)2  
 To: Personal Computer Users  
 From: Perfect Software, Inc.  
  
Subject: Features of @VALUE(SUM)3  
 @END(FLUSHLEFT)  
 @CENTER(-----)
```

Perfect Writer has the advanced features you would expect to find in expensive word processors, such as:

```
@BEGIN(ENUMERATE)4
```

<sup>1</sup> The STRING directive @STRING(X = Y) assigns the value of string Y to name X. This also works for assigning values from one name to another; if Y is not a quoted string, it is looked up in a list of names already set by the STRING directive.

<sup>2</sup> The BEGIN / END directives are used for environments which contain a lot of text. They ensure that, regardless of what standard fences are contained in the text, Perfect Writer will not prematurely finish an environment's formatting.

<sup>3</sup> The VALUE directive inserts the string previously assigned to the name inside the fences into the output document. @STRING and @VALUE are useful when the string to be inserted is used several times, is easy to misspell, or is very long.

<sup>4</sup> The ENUMERATE environment is used to make numbered lists of items or points to be mentioned. The ITEMIZE environment is used to make similar unnumbered lists. These can be nested within themselves or each other; nested ITEMIZE lists alternate between using a “.” and a “\*” for marking each item. The blank lines between each item are what tell Perfect Writer that it is time to start a new item.

---

The full range of @VALUE(SUM) document design options that allow you to automatically format letters, outlines, form letters, memorandum, articles, books or other types of documents.

Full cursor controls that allow you to move quickly and easily anywhere in the document in order to:

@ITEMIZE(insert

delete

search, search and replace, and search and replace with query)

@END(ENUMERATE)

What makes @VALUE(SUM) unique, however, are the features you won't find in other word processors costing many times its price, such as:

@BEGIN(DESCRIPTION)<sup>5</sup>

virtual memory@/Allows you to edit a full length novel as easily as you would edit a letter.

dual display@/Allows you to divide the screen into two parts, each showing a different document.

@END(DESCRIPTION)

We hope this very brief description encourages you to take a closer look at all the numerous features of @VALUE(SUM).

---

TEXT4.MSS

<sup>5</sup> The DESCRIPTION environment is used for just that—descriptions. It places the item to be described flush left to the margin and the description is heavily indented text on the right-hand side of the page. The "@/" command which appears in each description block makes Perfect Writer skip over to the description block.

---

## The Output:

To: Personal Computer Users  
From: Perfect Software, Inc.  
Subject: Features of Perfect Writer

---

Perfect Writer has the advanced features you would expect to find in expensive word processors, such as:

1. The full range of Perfect Writer document design options that allow you to automatically format letters, outlines, form letters, memorandum, articles, books or other types of documents.
2. Full cursor controls that allow you to move quickly and easily anywhere in the document in order to:
  - insert
  - delete
  - search, search and replace, and search and replace with query)

What makes Perfect Writer unique, however, are the features you won't find in other word processors costing many times its price, such as:

virtual memory Allows you to edit a full length novel as easily as you would edit a letter.

dual display Allows you to divide the screen into two parts, each showing a different document.

We hope this very brief description encourages you to take a closer look at all the numerous features of Perfect Writer.



## LESSON 5: Addressing an Envelope

This lesson will teach you about several commands which are useful when preparing documents not intended for 8.5 by 11 inch paper.

### The Input File

```
@STYLE(paperwidth = 9 inches)
@STYLE(paperlength = 4 inches)1
@STYLE(topmargin = 1 lines, bottommargin = 1 lines,
       leftmargin = 1 characters, rightmargin = 1 characters)2

@STYLE(headerspacing = 0 lines, footerspacing = 0 lines)3

@PAGEFOOTING( )4
@FLUSHLEFT(
Perfect Software, Inc.
123 Main Street
New York, NY 10007)

@BLANKSPACE(8 lines)5
@ADDRESS(
Robert Hemingway
1234 Main Street
Centerville, IL 60601)

@NEWPAGE6
```

---

### TEXT5.MSS

<sup>1</sup> The STYLE command allows you to set formatting parameters which are directly related to layout rather than descriptive of the text itself. The general format is @STYLE(parameter1 = value1, parameter2 = value2, ...). The style parameters we have set here have the obvious meanings. What we have done is to make most of the envelope useable for printing on.

<sup>2</sup> Parameters to commands which express distances are composed of an integer followed by a unit name. The units we've used here are inches, lines, and characters.

<sup>3</sup> Headerspacing is the distance between the header line and the start of the text, and footerspacing is the distance between the end of the text and the footer line. Even though the heading and footing lines do not appear, there is still space left, and we have just removed it.

<sup>4</sup> As you may have noticed, all the previous lessons were printed out with page numbers at the bottom of them. The way to turn off page numbering is to give an explicit pagefooting which is empty. This must be the very first command. [@PAGEFOOTING( )].

<sup>5</sup> The BLANKSPACE command is given a number of lines to skip.

<sup>6</sup> The NEWPAGE directive forces Perfect Writer to go to the next page immediately (in this case, the next envelope).

---

**The Output:**

Perfect Software, Inc.  
123 Main Street  
New York, NY 10007

Robert Hemingway  
1234 Main Street  
Centerville, IL 60601

---

TEXT5

---

## **LESSON 6: A Technical Paper**

Before we go on to this example, we need to introduce a new program called "Perfect Printer." Perfect Printer is used to print output which Perfect Writer's format program produces. Up to this point, we have been letting Perfect Formatter print the document as it formats it. From now on, you will probably use Perfect Printer to print things after Perfect Formatter has formatted them.

Why? Well, Perfect Writer's format program knows lots of things about text, but not very much about printers. So, if your document uses subscripts, partial line spacing, proportional-space type justification, or most of the other fancy printing features, you must use Perfect Printer.

Use Perfect Printer to print the following text. From the main menu select option F and enter "TEXT6.MSS" for the filename. From the Perfect Formatter Selection Menu, enter P and then G.

After the document is formatted you will need to print it. From the main menu select option P and enter "TEXT6" for the filename. From the Perfect Printer Selection Menu enter G.

Just as Perfect Formatter automatically assumes ".MSS," Perfect Printer assumes ".FIN." Include the "-pause" option on the command line if you want to hand-feed paper. This option stops at the end of each output page and waits for you to type any character on the console running Perfect Printer, so that you can prepare the printer and paper.

This example is a short excerpt from the first chapter in a book of technical papers. It is an overview of computer system security.

## The Input File:

```
@PAGEFOOTING(center = "@VALUE(Page)"1)
@CHAPTER(Computer System Security)2
@SECTION(Physical Security)3
@SUBSECTION(Computer Room Security)4
@INDEX(computer room security)5 There are three basic reasons
for maintaining computer room security.
These are:
@ENUMERATE[Preventing access to the computer itself.
```

Preventing access to the data physically contained in the computer.

Preventing access to the communication lines leading out of the computer.]

<sup>1</sup> The PAGEHEADING and PAGEFOOTING directives are used to specify the parts of the headings and footings for each page of the paper. There are three parts to each: the left, the center, and the right. You set each part as shown here. You may notice that the pagefooting we specified here is itself a Perfect Writer's format program command. The variable "page" is automatically changed at the start of each new page, so our footing will always contain the current page number.

<sup>2</sup> The CHAPTER directive causes all the bookkeeping involved with starting a new chapter to begin. Since this was the first chapter of the set, it was given chapter number one. An entry was automatically made in the table of contents containing the chapter title, number, and the page on which it began. A new page was started, and the chapter title was centered in a boldface heading.

<sup>3</sup> The SECTION directive does many things which are similar to the CHAPTER directive: it makes an entry in the table of contents, prints out a header (slightly less important-looking than the chapter header), leaves some blank space, and numbers the sections sequentially. Since this was the first section in the first chapter, it was given the number "1.1."

<sup>4</sup> The SUBSECTION directive is similar to the SECTION directive. It adds another number to the label, and does not underline the title of the subsection.

<sup>5</sup> The INDEX directive requests that an entry in the index be made, containing the text inside the fences and the page number on which the entry occurs. See example 1. Use of this command instructs Perfect Writer to produce an alphabetized index at the end of the document.

---

@PARAGRAPH(Computer access)<sup>6</sup>

There are simple reasons to keep people away from the computer itself:  
a vandal can  
physically destroy the equipment or disrupt the service it  
provides. Either can be costly, but both are easily discovered  
(after the fact!).

@PARAGRAPH(Physical data access)

Where there is a computer, there are usually disk and tape drives  
and the storage media they use in close proximity. Theft of  
online media is easy to detect (@I{most})<sup>7</sup>  
computer systems complain if you remove a disk or tape from the  
drive while it is in use!), but if the data contained therein is  
itself secret or valuable, detection is not as important as  
prevention. Theft of off-line media (e.g. backup tapes) is  
almost as easy to detect,...

---

TEXT6.MSS

<sup>6</sup> The PARAGRAPH directive is the lowest level of numbering available in documents. In particular, this paragraph would be numbered 1.1.1.1! Of course, you don't have to number your paragraphs—to get a plain new paragraph, remember, you just use a blank line in the document.

<sup>7</sup> The @I environment says to italicize the text inside the fences. Since most printers do not have italics, the text is usually underlined instead. To get boldface text, use the @B directive (i.e., @B{text to be boldfaced}).

---

**The Output:**

**Chapter 1**  
**Computer System Security**

1.1 Physical Security

1.1.1 Computer Room Security

There are three basic reasons for maintaining computer room security. These are:

1. Preventing access to the computer itself.
2. Preventing access to the data physically contained in the computer.
3. Preventing access to the communication lines leading out of the computer.

1.1.1.1 Computer access

There are simple reasons to keep people away from the computer itself: a vandal can physically destroy the equipment or disrupt the service it provides. Either can be costly, but both are easily discovered (after the fact!).

1.1.1.2 Physical data access

Where there is a computer, there are usually disk and tape drives and the storage media they use in close proximity. Theft of online media is easy to detect (*most* computer systems complain if you remove a disk or tape from the drive while it is in use!), but if the data contained therein is itself secret or valuable, detection is not as important as prevention. Theft of off-line media (e.g. backup tapes) is almost as easy to detect, . . .

---

TEXT6

---

Index

computer room security 4

**Example 1**

## Notes

---

## APPENDICES

---

### **In this Section**

---

- A. Installation Instructions
  - B. Error Messages
  - C. "Swapping..."
  - D. Glossary
  - E. Index
-

## **Notes**

## Appendix A **INSTALLATION**

### **Back-up Disks**

The first step in installing Perfect Writer is to make a copy or 'backup' of the original disks you received containing Perfect Writer.

This is important for two reasons: First, if you make a mistake while configuring Perfect Writer, your backup disk allows you to recover quickly with no damage to your original Perfect Writer disk. Second, if you are using Perfect Writer a great deal, the information on the original disk could become distorted and thus unusable. Having once made a backup copy of your disk, put the original away in some safe place, and always use the backup copy when editing with Perfect Writer.

### **Configuring Perfect Writer**

Configuration comes in two stages: configuring Perfect Writer to your **console**, and configuring Perfect Writer to your **printer**.

During configuration Perfect Writer will ask numerous questions not only about your computer and printer but about how you wish Perfect Writer to operate. Often you are asked to supply only a yes/no answer (i.e. "y" or "n" is sufficient). When Perfect Writer asks for one or more characters of information, type a carriage return after you have supplied that information. If you wish to accept the default values that Perfect Writer presents, type a carriage return and nothing more.

## **THE CONSOLE CONFIGURATION PROGRAM**

Insert your backup copy of Perfect Writer and call up Perfect Writer's "configuration program" by typing:

PWCONFIG followed by a carriage return.

The console configuration program, PWCONFIG, will lead you step by step through the process of configuring Perfect Writer to your console. Be sure to follow in order all of the instructions provided at the terminal console.

### **Specifying Your Console Terminal**

Perfect Writer asks you to specify the brand of computer terminal you own. Perfect Writer knows about several, and these are listed below.

## **Terminals Perfect Writer Knows About:**

- |                             |                               |
|-----------------------------|-------------------------------|
| 1 — Create new type         | 11 — BEEhive 150              |
| 2 — Heath H19               | 12 — Hazeltime 1500           |
| 3 — Televidio 912, 920      | 13 — Intertec Superbrain      |
| 4 — SOROC IQ120             | 14 — Processor Technology SOL |
| 5 — Lear-Siegler ADM-3A     | 15 — SCION Microangelo        |
| 6 — Lear-Siegler ADM-31     | 16 — Teleray 1061             |
| 7 — Perkin-Elmer Bantam 550 | 17 — Hewlett Packard 2621A    |
| 8 — Perkin-Elmer Fox 1100   | 18 — Hewlett Packard 2645     |
| 9 — D.E.C. VT-52 and VT 100 |                               |
| 10 — HDS Concept 100        |                               |

Numerous CP/M systems emulate (or act like) the above terminals. For instance, Lifeboat CP/M for the Radio Shack Model II emulates the "Lear-Siegler ADM-3A." Thus, if you are using Lifeboat CP/M for the Model II, then you would simply select terminal type "5". If you have an Apple computer you would define your terminal as a SOROC IQ120. Many terminals are like the ADM-3A, so if you are not sure about yours you might try this one.

## **Defining Your Terminal**

Skip this section if your terminal is listed and go to the section "Define I/O Port Characteristics." If your terminal is not listed as one Perfect Writer knows about, then you will need to select the "create new type" option. The configuration program then proceeds to ask you several questions about the operation of your terminal. Most likely, you will need to find the answers to these questions in your computer manual. Unavoidably the questions are detailed and technical and use terms that you may not be familiar with. If you are unable to answer these questions satisfactorily, it will be best to seek the help of a technician familiar with your computer and CP/M system.

---

*Appendix A-4 Installation*

---

---

## **TERMINAL DEFINITION QUESTIONNAIRE**

### **Questions Asked to Define a New Terminal**

1. Number of rows on your terminal? { 16 to 60 }
2. Number of columns on your terminal? { 64 to 132 }

---

Do you wish to enter the special terminal command codes in decimal, hex, or octal? { currently hex }
3. Does your terminal require a special initialization string to be sent to it at the beginning of the editing session?

If yes,  
Enter a list of hex numbers, separated by spaces
4. Does your terminal require a special deinitialization string be sent to it at the end of the editing session?

If yes,  
Enter a list of hex numbers, separated by spaces
5. Does your terminal have a command to clear the entire screen, leaving the cursor at "home" (the upper left corner)?

If yes,  
Enter a list of hex numbers, separated by spaces
6. Does your terminal have a special command to clear from the current cursor position to the end of the line?

If yes,  
Enter a list of hex numbers, separated by spaces
7. Does your terminal have a special command to clear from the current cursor position to the end of the screen?

If yes,  
Enter a list of hex numbers, separated by spaces
8. Does your terminal require any special characters to initiate a cursor positioning command (most terminals do)?

If yes,  
Enter a list of hex numbers, separated by spaces
9. Is the row or column to be sent first? { R or C }

10. Are the row and column codes sent in ASCII or Binary? {A or B}
11. Should the row and column numbers be complemented before sending? {Y or N}
12. Some terminals require that a bias be added to the row and column numbers. Usually this is 32 for both the row and the column. Enter 0 if none.

Row bias? (in decimal)  
Column bias? (in decimal)
13. Does your terminal require any special characters to separate the row code and the column code in a cursor positioning command?

If yes,  
Enter a list of hex numbers, separated by spaces.
14. Does your terminal require any special characters to terminate the cursor positioning command?

If yes,  
Enter a list of hex numbers, separated by spaces.
15. Does your terminal require any padding or delays to execute any of the special functions?
  - I. Padding can be done either by extra characters or by a delay loop.
    - a. Pad with characters?

If yes,
      - i. What is the code for the pad characters? {usually 0H, in hex; default = 0H}
      - ii. How many padding characters should be sent after a cursor positioning command? {default = 0}
  16. Does your terminal respond to the standard Ascii bell character, Control-G (07H)?

If no,  
What character should be sent to ring a bell or otherwise alert the operator?  
Enter a list of hex numbers, separated by spaces.

## **Answers to Terminal Definition Questionnaire**

Radio Shack TRS-80 Model II

Pickles and Trout CP/M

- |                |            |
|----------------|------------|
| 1. 24          | 9. row     |
| 2. 80          | 10. binary |
| 3. No          | 11. No     |
| 4. No          | 12. 32, 32 |
| 5. Yes CH      | 13. No     |
| 6. Yes 1H      | 14. No     |
| 7. Yes 2H      | 15. No     |
| 8. Yes 1BH 59H | 16. No     |

## **DEFINE I/O PORT CHARACTERISTICS**

After defining your terminal, you need to let Perfect Writer know if it is possible to use the CP/M BIOS for input and output. In most instances the answer is yes. You cannot use the BIOS for input, however, if your system has the so-called "common delete bug." In order to find out if your system does, Perfect Writer asks you to type a Delete (this key is sometimes labeled "Rubout," "Del," "back-space" or "◀" (left pointing arrow). If a "Y" appears on your screen, then you can use your BIOS; otherwise you will have to locate your hardware manual to find your input port characteristics.

Skip this section and go to "Your Individual Preferences for Perfect Writer" if you are able to use the CP/M BIOS. If you are **not** able to use the CP/M BIOS, then you will be asked several necessarily technical questions. They are:

## **DEFINING INPUT AND OUTPUT PORTS**

1. Use the CP/M (BIOS) input/output?
2. What is the number of the CP/M BIOS call to access this peripheral printer? Standard calls are:

3 — console input  
4 — console output  
5 — printer output  
6 — reader (modem) input  
7 — reader (modem) output

Your particular CP/M may have others defined.

BIOS call number:

3. Should the high-order bit of data bytes be masked out, leaving seven bits of data?
4. There are two ways to determine when a byte can be read from or written to a peripheral. CP/M BIOS calls exist to do this in some cases; in others, it is possible to directly read a status port. In cases where it is not possible to determine the status at all, answer "yes" to this question: Use CP/M BIOS for status determination?
5. What is the number of the CP/M BIOS call to test the status of this peripheral? The only completely standard call is 15, to test the console input status, but your CP/M may have others. In CP/M 2.0 and above BIOS call 15 is usually used for list status determination.

Enter 0 if none exists.

BIOS call number:

6. A mask will be ANDed with the status byte (from either a port or the BIOS) to get a zero or non-zero value, depending only on whether the port is ready to send a character (for output ports) or has a received character ready (for input ports). If a BIOS call is being used to get the status, this should usually be FF hex; otherwise, it will be a byte of zeroes except for a single 1 bit. Ready mask (in hex):
7. Does ANDing the status byte with the ready mask give a non-zero value when the port is ready to send a character (for output ports) or has a received character ready (for input ports)? If a BIOS call is being used to get the status, the answer is usually "yes." Ready when non-zero?

## Your Individual Preferences for Perfect Writer

There are several questions in the configuration program to which there are no correct answers. Rather, they concern your personal preferences.

- The first such question concerns the size of the **Swap File**. Perfect Writer's 'virtual memory' capability, which allows you to write and edit extremely long documents, is made possible by a unique programming feature called the 'swap file.' This is simply a storage space (a file) where Perfect Writer places parts of your document which you are not currently editing. The filename is "PW.SWP."

The size of this swap file is important, because it limits the total amount of text you can be editing at one time. A workable minimum is about 24 pages, while the maximum size the swap file can be is 248 pages. Roughly a 64-page swap file lets you edit documents up to 64K bytes in size. Perfect Writer is provided initially with a swap file of 64K bytes, a size which you can change during the configuration process.

- The second question concerns the line on which the cursor is initially displayed. The default value for this line is just above the center of the screen. This is where the cursor will be displayed following any change or redisplay of the screen, such as is produced by the VIEW NEXT SCREEN, VIEW PREVIOUS SCREEN, or REDISPLAY SCREEN commands.
- The third, fourth, and fifth questions concern the values for tab spacing, and the left and right margins. Though Perfect Writer comes with these values predefined, you can change them during configuration, or at any time while editing.
  - \* Tab spacings are pre-set at eight column intervals. That is, the first tab is at column eight, the second at 16, the third at 24, etc.
  - \* The right margin, or 'fill' column as it is sometimes called, is set initially at column 65. This represents the last column in which characters can appear. Perfect Writer automatically shifts text as you type so that words do not extend beyond this column.
  - \* Related to the right margin is the 'indent' column, which is essentially the **left margin**. It is the first column on the screen in which text can appear. It is initially set at **zero** and causes text to begin at the left margin.

- The sixth question concerns the 'delay count.' When you are not working at the keyboard, Perfect Writer will take the opportunity to do its internal 'housecleaning chores,' such as exchanging portions of your text to and from the swap file. The delay count tells Perfect Writer how long to wait after you have stopped working before proceeding with its own internal chores. The delay count is initially set at four seconds. This is achieved by entering a value of 300 during configuration. Doubling or halving that value, doubles or halves the delay.
- Next, Perfect Writer asks the clock rate of your computer's microprocessor. This is usually specified in megahertz or MHz. Typical rates are two or four MHz. If you are unsure of your computer's clock rate, then select default value by hitting a carriage return.
- Finally, Perfect Writer allows you to swap two pairs of keys or characters on input from the keyboard. This feature can be used to overcome many deficiencies in keyboard design and layout. For instance the 'Delete' character is a very useful character to be able to type. Yet, on many terminals it is a shifted character, that is, one must hold the shift key down to type 'Delete'. Perfect Writer can help you overcome this by allowing you to interchange the 'Delete' character and the other character on the same key. This feature can also be used to swap the key of a character on your keyboard with a command character that you do not have a key for. After swapping keys Perfect Writer will interpret the swapped key as the command you have assigned it. This allows you to convert keys on your keyboard that you might not otherwise use into the keys you do not have but need. For example, if your terminal does not have a "delete key" but has a "backspace" key, then you should swap the "delete command" [Hex=7F] with the "backspace" key [Hex=08] and convert the "backspace key" into a "delete key."

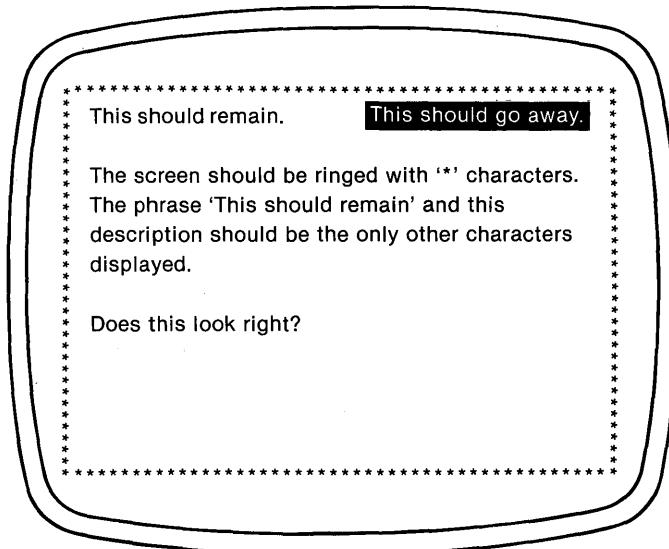
## Testing

After you have completed the configuration program for your console terminal, you should select option six: **terminal testing**.

This test should automatically clear the screen, and print columns of asterisks, '\*', along the left and right edges from the bottom up. Then asterisks should appear along the top and bottom edges of the screen, beginning at the left and ending on the right.

---

The words "This should remain" and "This should go away" will appear on the screen near the top, except that the latter should be erased almost immediately. The screen display should look like this:



If the asterisks do not properly ring the screen, either the cursor positioning sequences were performed improperly, or the size of the screen was not specified correctly.

If the text "This should go away" remains on the screen, the 'clear to end of line' function is probably not working. If any characters are missing, or if there are additional random characters, the 'padding' might be wrong.

If the terminal test rolls the screen up one line, try reducing the number of columns by one (via option five) and rerunning the test. If the test now works you can restore the number of columns to its previous value, and Perfect Writer should work.

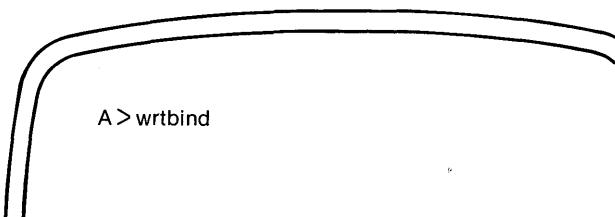
Don't forget to install your terminal definitions (option seven) on the swap file so that Perfect Writer will be able to find them later.

---

## RUNNING THE BINDINGS PROGRAM

All of the editing commands in Perfect Writer are executed by typing command keys. For instance, to move the cursor back one character you type Control ---- B. What this means is that the BACKWARD CHARACTER command is bound to the Control ---- B key combination. Perfect Writer offers a revolutionary new feature not found on any other word processor. With Perfect Writer you can change the command key bindings to fit your particular needs and desires. To change the bindings you need to run the bindings program. In the discussion that follows we will explain how to implement the conventional bindings supplied with Perfect Writer. We recommend you begin with the conventional bindings (which are documented in your User's Guide). Later, if you decide to change the command keys you will need to edit the bindings table and run the PWBIND.COM (the bindings editor) program discussed in the section entitled, "Changing the Command Keys."

After installing Perfect Writer on your console (using PWCONFIG.COM), you are ready to run the bindings program (WRTBIND.COM). If you want to use Perfect Writer exactly as it is described in the User's Guide, simply run the WRTBIND.COM program. The discussion in the next section explains how you may change the command keys in Perfect Writer to fit your particular desires. We recommend that you begin using the standard Perfect Writer and only change the command keys after you have used the standard Perfect Writer a while and have come up with some changes you are confident will improve the program. To run the bindings program make sure that both PW.SYM and PW.SWP are on the same disk as WRTBIND.COM. Then simply enter:



WRTBIND.COM writes out FUNCTS.TXT (a table of binding-to-key matches), FUNCTS.DSC (a table of descriptions of each of the available functions), and writes the bindings themselves into PW.SWP (the swap file).

This completes the console configuration procedure.

## **CHANGING THE COMMAND KEYS**

**Note:** You only need to read the following text if and when you want to alter the command keys used by Perfect Writer. If you are using Perfect Writer for the first time, then you may want to skip this section until later when you want to change the editor.

By itself the WRTBIND.COM program discussed above does not allow changing the command key bindings. Its use is intended for those who are satisfied with the 'standard' bindings arrangements, or for recovery from failure to successfully change the command bindings with PWBIND.COM.

PWBIND.COM requires FUNCSTS.TXT, PW.SYM and PW.SWP (which it modifies). The initial installation of Perfect Writer is accomplished with WRTBIND.COM as discussed above. This first Perfect Writer can then be used to edit FUNCSTS.TXT to reflect your desired command structure arrangement. Once FUNCSTS.TXT is edited, you simply install these bindings by running PWBIND.COM.

After configuration is complete, only PW.COM and PW.SWP need to be present for Perfect Writer to run.

## The FUNCTS.TXT File

FUNCTS.TXT is a specially constructed file. If things in the file are taken out of order, or parts of it are erased, or it is damaged, then errors are almost sure to follow. Because it is easy to improperly edit FUNCTS.TXT, we have included the program that generates it: WRTBIND.COM. If you improperly edit FUNCTS.TXT and cannot figure out what is wrong, you should probably start all over again with a fresh copy. WRTBIND.COM will generate one for you.

FUNCTS.TXT is meant to be edited with the tab stops set to eight, the indent column set to zero, and the fill column set to the maximum width of the screen. It is meant to be edited in Normal Mode. (Most crucial is the tab setting. If it is not eight, then the file will look 'funny' and the columns will not line up. Inserting extra spaces in such a case to make it look 'not funny anymore' will damage the file.)

Since it is crucial that FUNCTS.TXT be correct in every detail, PWBIND.COM makes numerous checks to insure the integrity of FUNCTS.TXT while it is installing your new bindings. If there is a flaw in FUNCTS.TXT, PWBIND.COM will report the flaw(s), and will abort without altering the bindings in Perfect Writer.

If this sounds ominous and scary, that's because it is. It will be easy for you to make errors with PWBIND.COM and FUNCTS.TXT files. Your first attempt at random bindings is likely not to be successful. Fortunately, you can always get back to the 'standard' with WRTBIND.COM. Make sure that you do not experiment with any of these things on your distribution diskette. Make two backup diskettes first, and then save your distribution diskette and a backup in a safe place.

FUNCTS.TXT consists of three header lines and 384 data lines. All of these lines are critical to the operation of FUNCTS.TXT and the misordering or deletion of any or the insertion of extra ones will result in an error message from PWBIND.COM (see Appendix A-43 to A-52 for a listing of the FUNCTS.TXT file).

---

The first header line in FUNCTS.TXT is a copyright notice. This notice must remain in the FUNCTS.TXT file and it cannot be altered in any way.

The second header line in FUNCTS.TXT contains information about the default mode. PWBIND.COM will check for a valid mode name as the first word on this line. If there is not a valid mode name in the first word position, then PWBIND.COM will set the default mode to 'Normal' and continue. There can be no leading spaces or tabs before the word, and there must be at least one space after it.

The third header line in FUNCTS.TXT are headings for the four columns in the data lines. This line must remain in the FUNCTS.TXT file and cannot be altered in any way.

Each data line in FUNCTS.TXT consists of four columns. The information in each of these columns has meaning, and you should not alter this information unless you are sure you know what it means first. Each column must be separated from the next by a single tab character. Each line must be separated from the next by a single newline character (that is, the file is single-spaced vertically).

1. The first column in each data line in FUNCTS.TXT is an ordinal number. The number is in the range {0...383} and is the number of the function key associated with that line. The ordinal numbers in the first column must remain in the order that they are in now, that is to say, they must remain in numerical sequence. If the numbers in the first column are taken out of sequence, or if a non-number is found in the first column, an error message will result.
2. The second column in each data line in FUNCTS.TXT is a textual representation of the key that is associated with that line. Control commands are prefixed with a 'C'. Meta commands are prefixed with a 'M'. EXTended commands (Control-X) are prefixed with a 'X'. Combinations that produce valid commands can be formed and are also represented (e.g. M-C-O and X-C-F). Since the prefix commands themselves can be altered with the facilities here, commands will not be referred to as 'Escape' or 'Control-X' commands.

3. The third column in each data line in FUNCTS.TXT contains the name of a function that is bound to the key associated with that line. The names are truncated to eight characters. The name in this column must be a name of a function that is listed in the FUNCTS.DSC file. If it is not, an error message may result, or Perfect Writer may not operate properly. The name must be spelled correctly, and must be in upper case. More than one key can be bound to the same function. This is evidenced by having the same name on many different lines in the third column.
4. The fourth column in each data line in FUNCTS.TXT contains a short description of the action performed by the function on that line, or the name of that function, if it has one. This information must be present, or an error will result. However, the actual information itself is ignored, once its presence has been checked for. It is recommended that if any changes are made to the third column, that the fourth column be updated accordingly.

Therefore, it follows that the only things in FUNCTS.TXT that can be changed are the first word of the second line, which controls the default mode, and the third and fourth column in each of the data lines, which control the function bound to the key associated with that line. If any other things are altered, an error message may be produced, or Perfect Writer may fail to perform as expected.

Again, if you get into trouble, run WRTBIND.COM and start all over. It is easier to begin again than to try to untangle a mess. It is often helpful to print out FUNCTS.TXT and plan the changes you intend to do on paper first, before actually tackling the task. This is, in fact, what we recommend.

## An Illustration of Editing the Command Keys

In this section we will give some examples of how you can change some things about. We will describe some of the pitfalls that are not obvious at first glance. Print out a copy of FUNCTS.TXT and a copy of FUNCTS.DSC and have them handy as you continue reading below.

If you wanted to bind EVERY command key to the function that beeps the bell and prints 'Unknown command', then you would replace the function name in the third column of every line, and the description in the fourth column with:

MNOTIMPL      Used for keys which are not used.

This would not be a very useful modification to Perfect Writer. Nonetheless, it is one that PWBIND.COM allows. PWBIND.COM is unable to check to make certain that you are not doing something silly, it only checks to see that you are not doing something patently wrong.

On the other hand if one wanted to rebind the 'Center Line' function so that it resided on the X-C-C key (for 'Center') instead of the M-S key, one would need to place

MCNTERLIN      Center Line

in the third and fourth column of line 258. (Note that M-s and M-S are distinct, and can have different functions bound to them.) PWBIND.COM will allow this change, however we have now replaced the QUIT command with another one, and if we begin Perfect Writer we will not be able to quit, short of 'pulling the plug.' Again, this is a problem, but it is a problem in a subtle way. To fix this problem we need to bind the QUIT command to another key. X-C-E (for 'Exit') is a good candidate, so on the lines that X-C-E are associated with you should place

MEXIT      Exit Perfect Writer

in the third and fourth column. Can you figure out what line this will have to be by looking at FUNCTS.TXT?<sup>1</sup> If we do not want to use M-S and M-s any more, can you figure out how we will make them 'Unknown commands'?<sup>2</sup>

1. It would have to be on line 261.
2. We would place the following on lines 211 and 243:

MNOTIMPL      'Unknown command'.

---

Some terminals have function keys that generate an Escape character followed by another character. If the other character is an upper case alphabetic character (as it is for the H-19) then one can leave the M-lowercase commands as they are and have the M-uppercase commands do something different. This is because most users type with the shift lock key up so that M- commands that they type are all M-lowercase commands. The computer can tell the difference, though and treat the upper and lower case differently. Suppose that the F1 key generates an Escape followed by an uppercase 'P'. In the usual Perfect Writer, M-P and M-p are both 'Up Paragraph'. However, since we usually type M-p, and rarely if ever type M-P we would like to use the F1 key on your terminal for something other than 'Up Paragraph'. We would like F1 to start a 'Save File'. We can do this by placing

MFILESAV Save File.

on the line that M-P is associated with in the third and fourth columns. Now when we type M-p we get 'Up Paragraph', and when we hit F1, we have started a SAVE FILE command, and now can answer the prompt.

Since the prefixing process is also done with commands, you can change the prefix for Meta or eXtended commands. Suppose that you would prefer that the prefix for eXtended commands was C- instead of C-X. You can do this by placing the MCTRLX command on the line with C- and moving MTOGLC to another key. Can you figure out which lines must be changed?<sup>3</sup>

Some people are familiar with the use of the arrow keys on their terminals. Since nearly every terminal does something different when an arrow key is depressed, and since it is almost never mnemonic, when designing Perfect Writer we elected to make the entire command set as mnemonic as possible and to ignore the arrow keys. If you feel like you must use the arrow keys, you can rearrange the command set so that the arrow keys do whatever it is that you think they should. However, since most of the characters that the arrow keys are likely to generate are probably already bound to something mnemonic, you may find that you will need to rearrange much of the command set so that you can use the arrow keys. Further, the result may no longer be mnemonic. If you really want to install the arrow keys, plan it on paper first, by printing out FUNCTS.TXT and writing your intended changes in the margins until you get it all the way you want. Don't forget that if you want to use a command you must bind it to a key that you can type.

---

3. Line numbers 24 and 30.

We believe the commands-to-keys relationship that we have provided is quite versatile. On most terminals all of the commands in Perfect Writer can be typed without the typist having to move his or her hands away from the home position on the keyboard. Our experience, and the volume of academic research shows that the time it takes to use an off-home key, like an arrow key or a function key is always greater than the time it takes to type even a two or three key command that is reachable from the home position. One estimate is that as much as three to five seconds can be lost hunting for an off-home key, even for an experienced user. If you are a slow typist, it is an irritant. If you are a fast typist it is a major hindrance to have to go to an off-home key.

You will have to determine for yourself what you like best. You may prefer to use off-home keys, even though you can see that it slows you down. That's fine. Or you may set up all the function and arrow keys only to find out that you liked it better the way we delivered it. That's fine, too. Since we cannot determine your tastes in advance, and certainly would not presume to do so, we have provided what WE like best, and the ability for you to make it into what YOU like best.

## **CONFIGURING PERFECT WRITER TO YOUR PRINTER**

Call up Perfect Writer's printer configuration program: "PFCONFIG," by typing "PFCONFIG" . . . followed by a carriage return. Like Perfect Writer's "PWCNFIG" program, this program will lead you step by step through the process of configuring Perfect Writer to the particular capabilities of your printer.

### **The PF.DAT File**

Perfect Writer will store all of the data that you enter regarding your printer in a special file called the "PF.DAT" file. When formatting documents it will consult this file for the specifications you supply during this configuration procedure.

### **The Default Printer Configuration**

Perfect Writer comes preconfigured for a simple, 'no frills' printer which is known to Perfect Writer as "Vanilla." The "vanilla" printer cannot do underlining, boldfacing, proportional spacing, or anything else that is complicated. Your printer may do more than the "vanilla" configuration. However, try using Perfect Writer with this default configuration first, returning afterwards to the configuration procedures to more adequately describe your printer to Perfect Writer.

## **Answering the Configuration Questions**

As with the console configuration program, yes/no questions can be answered by typing simply "y" or "n". Questions that require you to enter one or more characters of information must be followed by a carriage return to tell Perfect Writer that you have finished entering the information.

You will be entering the following types of information while configuring your printer to Perfect Writer:

1. Decimal numbers, between 0 and 65535 (mica values are entered in decimal).
  2. Hexadecimal numbers, between 0 and FF (used in the I/O port descriptions). The hex digits A-F may be entered in either upper or lower case.
  3. Character strings, which are entered just by typing the characters (not in hex). To enter a carriage return (^ M, 13 decimal), Delete (Del, ?, 127 decimal), Control-Q (^ Q, 17 decimal), Control-G (^ G, 7 decimal), or back-space (^ H, 8 decimal), type a Control-Q before the special character you wish to enter.
- Control-G is the PFCONFIG universal cancel command. You can type it in answer to any question, either to skip the question and move on (the value being requested remains unchanged) or to quit doing something.

## **What a Device Is**

To Perfect Writer, a 'device' is not a physical machine, but rather a definition of how any particular physical printer is to perform in a printing situation. All aspects of a printer are part of this definition, including apparently extraneous items, such as the size of paper it will use and the type of printwheel.

A single physical printer can be defined to be more than one device. For example, it can be defined to handle two different kinds of paper, such as mailing labels and standard paper. Each definition constitutes a different 'device' to Perfect Writer.

## **Types of Output**

Perfect Writer can generate four types of output. These types are:

1. direct to console
2. direct to printer
3. to a standard system file
4. to an intermediate file

The first three types of output are intended for, respectively, quick draft output to a terminal screen, draft output to a printer, and output to a standard system text file. These first three do not include subscripts, superscripts, boldface, underscoring, or any special character attributes. No printer control is allowed for. No vertical or horizontal microspacing is allowed, nor is microspace justification performed. Proportionally spaced devices cannot be defined for these types of output.

The fourth output type, however, deals with everything. With it, Perfect Writer outputs to an intermediate file (with the suffix ".FIN"). The printing program is then used to print the final copy.

## **All Measurements in Micas**

Perfect Writer performs all internal measurements in micas. A mica is a unit of distance equal to 10 microns or 1/100,000 meters. It is a small enough unit that any distance used in formatting can be described accurately. Here are some conversions and measurements:

$$\begin{aligned}1 \text{ mica} &= 10 \text{ microns} = 10 \mu\text{m} \\&= 1/100 \text{ mm} \\&= 1/1000 \text{ cm} \\&= 1/100,000 \text{ m} \\&= 1/2540 \text{ inches}\end{aligned}$$

(the following assume 10 characters/inch and 6 lines/inch)

1 character = 254 micas  
1 line = 423 $\frac{1}{3}$  micas

1 inch = 2540 micas  
 $\frac{1}{2}$  inch = 1270 micas  
1 centimeter = 1000 micas  
1 pica = 423 $\frac{1}{3}$  micas  
printer's measure =  $\frac{1}{6}$  inch = 12 points  
1 point = 35  $\frac{5}{18}$  micas  
printer's measure =  $\frac{1}{2}$  inch

## Paper

8 $\frac{1}{2}$  inches = 21,590 micas  
11 inches = 27,940 micas  
14 $\frac{3}{4}$  inches = 37,465 micas  
  
21 cm = 21,000 micas  
30 cm = 30,000 micas

## Mailing labels

30 characters = 7620 micas  
6 lines = 2540 micas

## Terminal screen size

80 characters = 20,320 micas  
24 lines = 10,160 micas

Micas are always positive numbers. However, values greater than 48,000 micas are not allowed. Micas are used for describing paper measurements to Perfect Writer.

Whenever computing the size of a terminal screen you should assume that it prints 10 characters per inch and six lines per inch and then compute the effective paper size based upon the size of the display. Most video terminals cannot, of course, microspace or microfeed.

---

## Selecting and Defining Ports

PFCOMFIG's "ports" tell Perfect Writer how to do the physical input and output on your computer system. Peripherals can be accessed either as CP/M operating system devices or through direct hardware port input/output connections.

Perfect Formatter and Perfect Printer come with a configuration file (called "PF.DAT") which has preconfigured PFCOMFIG "ports" for console Input/Output and printer output. You will not need to change the console port definitions. If your printer is configured as the system listing device (You can tell that it is if you can type a Control-P on the console to get output (such as a directory) echoed to the printer.), you need do nothing: the default printer port, "Printer Out," uses the CP/M listing device.

## Synchronization Protocol

If you are using a printer which does not require a synchronization protocol [(X-ON/X-OFF) (Control-S/Control-Q) or ETX/ACK (Control-C/Control-F)] then skip to the section "Printers That Perfect Writer Knows About." Otherwise, see your printer manual. Your computer must listen for input back from the printer as well as sending characters to it. Synchronization protocols work by having the printer send characters back to the computer to say when it is and isn't ready; this is because the computer can usually send characters faster than the printer can accept them.

If your synchronization-protocol printer is the system listing device, it is possible that the synchronization protocol is automatically handled by your system. If you are running a version 2.x CP/M system, you may have to set the default output port to "Printer 2.2" instead of "Printer Out." In either case, make sure that you do **not** have the printer type defined (see the next section) to use the synchronization protocol; the protocol referred to in the printer type definition menu is a "manual" one which Perfect Printer will do if your CP/M does not automatically provide it.

---

If you discover that your printer loses characters from Perfect Formatter or Perfect Printer, chances are that your BIOS is not perfectly configured for your type of printer. If this is so, you will have to do direct hardware port input/output to it. The input port is for the printer to tell the computer to wait for it to catch up. (For this case, you **should** select a synchronization protocol in the "Define Printer Types" menu.) Enter all the port definition parameters (e.g. port addresses, status masks, etc.) as they are described by the "Enter New Port" or "Update Existing Port" menu selections of the PFCONFIG program. (Note that the update menus have briefer descriptions than the entry menus—you may want to define the new "Printer In" port first in order to get the longer explanations before actually updating the "Printer Out" definition.)

Using the BIOS to automatically take care of any synchronization protocol which the printer requires may have a penalty in speed: during the time that the BIOS is waiting for the printer, Perfect Printer could be setting up more characters to print. The problem is that Perfect Printer doesn't know when the printer is ready, and so has to call the BIOS as soon as each character becomes available. If you have a serial interface, therefore, even though it's easier to use the simple "Printer Out" CP/M listing device port, you may want to directly access the hardware ports and use a synchronization protocol; if a parallel, you may want to look directly at the port's ready bit.

If you have more than one printer attached to the system at once, you will have to define separate PFCONFIG "ports" for each additional hardware port or CP/M device you want to use. For example, you may want to use the "modem" or "punch" ports for your second printer. As with the printer port, your system may take care of all the Input/Output work associated with the device, in which case you can specify the standard BIOS calls when defining the PFCONFIG "port." If not, you may have to specify the actual hardware port numbers and perhaps define an input port for any synchronization protocols, as explained above. If you must specify an input port for any output port, the names of the associated pairs of output and input ports must be of the form "XXXXXX Out" and "XXXXXX In." This will allow you to specify just the first portion of the name when using the "-port" option on command lines.

## **Printers that Perfect Writer Knows About**

This section presents a brief discussion of the printers that Perfect Writer knows about. You can define your own printer type if it isn't one of the ones which already has set-up data. It should be noted that there can be more than one device description for a single type of printer. Printers often have different print widths (for example, many print both 10 and 12 pitch). Perfect Writer can deal with many different devices, but not more than one in the same document. Generally, you will have one width as the default (say, 10 pitch), and then have a second device handle the other width. This is done cleanly by having the initialization string for the second device put it into, say, 12 pitch mode and the termination string bring it back to 10 pitch.

On many printers, the paper can be moved horizontally relative to the print positions. Perfect Writer assumes that the first print position is at the left edge of the sheet of paper.

The Perfect Printer program (called "PP.COM") tries to perform full print head movement optimization and so it is worthwhile to describe your printer as accurately as possible. In addition, it is generally worthwhile to turn off any hardware "smarts" (e.g. automatic five lines before end of page formfeed) your printer may have.

Perfect Printer goes to a great deal of trouble to get everything just right, and any fanciness which your hardware interjects is likely to cause more problems than it solves.

## **Very Simple Printers**

The "vanilla" device is generic for all very simple printers. It assumes that the printer cannot back up either by using backspace or a bare carriage return. Perfect Formatter comes with this as the default device. If you have any but the simplest of printers you will want to change the initial default to something more appropriate.

---

## **Printers That Can Back Up**

The "plain" device is generic for most printers. It assumes only that the printer can back up by using a bare carriage return. This device is suitable for a wide variety of printers. For printers that can backup using the backspace character this device can be simply edited to add or substitute the backspace capability.

## **The Epson MX-80**

The "Epson" device is used with the Epson MX-80. This printer can microfeed, but it cannot microspace. Therefore, it can do super- and subscripts well, but not microspace justification. Perfect Formatter can deal with any of the character widths, but the width must be the same for the entire document. Perfect Formatter cannot use enhanced printing modes on parts of a document because they always affect the entire line.

Most versions of the MX-80 come with a paper tear bar on the cover. The initial paper offset defined for this device comes set up to have the top of form line up with the tear bar, but the offset might be different for the form's tractor version.

If your MX-80 has a serial interface, you will have to specify the X-ON/X-OFF (Control-S/Control-Q) synchronization protocol. With the parallel interface, the port must be defined to do hardware handshaking.

## **Diablo and Xerox**

The "Diablo10," "Diablo12," and "DiabloPS" devices cover the Diablo 1610, 1620, 1640, 1650, 630, equivalent Xerox models, and any other printers compatible with one of these. These models can both microfeed and microspace. Therefore, super- and subscripts work well, as do microspace justification and proportional typewheel printing. Text will be underscored with solid lines. Bidirectional printing will be done on this printer.

These devices assume X-ON/X-OFF (Control-S/Control-Q) synchronization protocol; some of the earlier models use ETX/ACK (Control-C/Control-F) protocol instead and you will have to change the definition. You must be sure to define the printer **input** port in order to use either protocol with this device.

These printers are capable of 10 pitch, 12 pitch and proportional spaced printing. The "Diablo10" device is for 10 pitch, "Diablo12" for 12 pitch, and "DiabloPS" for proportional space typewheels.

## **The NEC Spinwriter**

The "Spin10," "Spin12," and "SpinPS" devices cover the Nippon Electric Company (NEC) 5510 and 5520 printers. These models can both microfeed and microspace. Therefore, super- and subscripts work well, as do microspace justification and proportional typewheel printing. Text will be underscored with solid lines. Bidirectional printing will be done on this printer.

The printer requires X-ON/X-OFF (Control-S/Control-Q) synchronization protocol. You must be sure to define the printer **input** port before using this protocol with this device.

These printers are capable of 10 pitch, 12 pitch and proportional spaced printing. The "Spin10" device is for 10 pitch, "Spin12" for 12 pitch. The proportional print thimble has a non-ASCII arrangement of characters and requires the use of a translation table.

The pitch switch on the front panel must be set to "10 pitch" for 10 pitch printing and "12 pitch" for 12 pitch or proportional spaced printing.

---

## **The Integral Data Systems 460**

The "IDS460" device is used with the Integral Data Systems 460 and 465 printers. This printer is capable of proportional spaced printing and microspace justification but we do not recommend such use because attempting this with Perfect Formatter will cause the print head to make a separate pass for each word (and a lot of noise).

## **Centronics 737**

The "CENT737" device is used with the Centronics model 737 printer. This printer is capable of proportional spaced printing and microspace justification.

## **CP/M File Output**

The "File" device comes set up for a standard system text file. It is 80 columns wide by 66 lines (11 inches) high. Output when using this device will be readable by programs under CP/M but not suitable for use with Perfect Printer.

## **The Console Device**

The "Console" device is the initial default for use with the "-c" option to Perfect Formatter, and is intended for draft output to a terminal. This device comes set up for use with a 24 by 80 CRT terminal. When describing files or terminals, you should assume print widths of 10 characters per inch and six lines per inch to reduce confusion. The sheet of paper then becomes 4 by 10 "inches," regardless of the physical screen size.

## **Any Other Kind of Printer**

If the configuration program's printer selection menu does not mention your printer or a suitable substitute, you need to select the "create new printer" option (option 1). PFCOMFIG proceeds to ask you lots of detailed questions about things that you might never have heard of. If you know the answers, fantastic! If not, try to look the answers up in your printer manual, find a local expert, or contact your printer manufacturer.

---

## **PRINTER DEFINITION QUESTIONNAIRE**

### **Questions Asked to Define a New Printer**

1. Name of printer to be defined:
2. What is the width of paper? [1 inch = 2,540 micas; 8½ inches = 21,590 micas] Width in micas:
3. What is the height of paper? (1 inch = 2,540 micas; 11 inches = 27,940 micas) Height in micas:
4. What is the standard character width? (On a fixed width device, the width of any character; on a proportionally-spaced device, the width Perfect Formatter will use when distances are specified in characters. Capital 'O' is a good one to use.) [10 characters per inch: 254; 12 characters per inch: 212; 16.5 characters per inch: 154] Width in micas:
5. What is the height of a simple-spaced line? [6 lines per inch: 423; 8 lines per inch: 317] Height in micas:
6. What is the smallest horizontal movement the printer can make (i.e., the horizontal resolution)? [10 pitch: 254 micas; 12 pitch: 212 micas; 16.5 pitch: 154 micas;  $\frac{1}{120}$  inch microspacing: 21 micas] Width in micas:
7. What is the smallest vertical movement the printer can make (i.e., the vertical resolution)? [6 lines per inch: 423; 8 lines per inch: 317;  $\frac{1}{48}$  inch microspacing: 53] Height in micas:
8. Does your printer have a proportionally spaced font or printwheel that you intend to use for this device? Yes or No:

*Appendix A-32 Installation*

---

## Width and Translation Tables

A special feature is used to support the Spinwriter: translation tables. The character layout on the Spinwriter PS thimbles is non-standard, so we use a translation table to determine what character to send to the printer to get the one we want to print. These tables are, of course, in PF.DAT and are entered and edited with PFCONFIG.

PF.DAT also has character width tables for the Centronics and for typical Diablo printwheels and Spinwriter thimbles; you may wish to edit either of the tables to be more suitable for your printwheel or thimble. So, of the printers that Perfect Printer can drive and which are capable of proportional printing, the only one on which it is not fully supported is the IDS 460. (The IDS can do it, but it will print a word at a time, reversing direction and making noises between each one.)

The supplied PF.DAT contains the following width and translation tables:

- |         |   |
|---------|---|
| Width 0 | A 10-pitch fixed-width table. The width of the delete character (ASCII DEL, decimal 127) tells Perfect Printer how much extra space to insert at the ends of sentences; if you want to change that amount, but otherwise use a fixed-width 10-pitch font, you can do that with this table. Also, if you want to generate a width table for a new proportional font, a fixed width table is a good starting point. |
| Width 1 | A table for the "Bold PS" Spinwriter thimble.   |
| Width 2 | Official Diablo proportional-spacing widths. You may want to change these slightly to reflect your printwheel.  |
| Width 3 | Widths for the Centronics 737 proportional-spaced font.   |

Translation 0 Some Diablo printwheels and Spinwriter thimbles have a slightly non-ASCII arrangement of characters. This one is for a particular Spinwriter thimble; all but the "#" and " " characters translate into themselves. You will find it a convenient starting point if you require a similar rearrangement.

Translation 1 This works for some proportional spacing Spinwriter thimbles and printer models. You may have to change it. In order to get some shifted characters (the ones on the bottom of the spokes) to be printed, the eighth bit in some characters is turned on. (In the PFCONFIG display, this is shown by a tilde preceding the character.) Turning the eighth bit on is a sentinel which Perfect Printer recognizes; it makes Perfect Printer send appropriate "shift out/shift in" control sequences to the Spinwriter.

## A Question of Taste and Style

There are several questions that the configuration program asks you for which there are no "correct" answers; rather, they are about your personal preferences. The answers given here will be the defaults that Perfect Writer will use on your system. They can, however, be overridden in any given document by use of the STYLE directive (see Chapter XVIII, pages 2-6).

The questions are:

Top, Bottom, Left, and Right Margins	Entered in micas. These entries indicate the amount of the page that Perfect Writer will not print on. The top and bottom margins are initially 1,270 micas ( $\frac{1}{2}$ inch) and the left and right margins are initially 2,032 micas ( $\frac{8}{10}$ inch = 8 characters).
Headerspacing	Entered in micas. This entry is the amount of blank space to leave between the end of the header and the beginning of the running text on the page. This entry is initially 1,270 micas ( $\frac{1}{2}$ inch or 3 lines).
Footerspacing	Entered in micas. This entry is the amount of blank space to leave between the end of the header and the beginning of the footer. This entry is initially 1,270 micas ( $\frac{1}{2}$ inch or 3 lines).
Linespacing	Entered in tenths of a line. This entry is the amount of space to leave between each line in those environments that don't explicitly say otherwise. This entry is initially 10 (1 line).
Spread	Entered in micas. This entry is the amount of additional space to leave between paragraphs. This amount is added to the linespacing entry to determine the actual amount of space left. This entry is initially 423 micas (1 line).

Indent	Entered in micas. This entry is the amount to indent the first line of paragraphs in QUOTATION and TEXT formats and the amount to indent the body of paragraphs in UNDENT. This entry is initially 508 micas (2 characters).
Tab Spacing	Entered in units of characters. Perfect Writer has semi-variable tabbing: tab stops are set automatically a fixed number of columns apart (for example, if the spacing is eight, tabs will be set at columns 8, 16, 24,...). ANSI (American National Standards Institute) does not have a standard for this, but recommends an eight column spacing; this is also the CP/M standard. We prefer a five column spacing, however. (Note that tab characters in input are treated as spaces, except in environments such as VERBATIM that conserve whitespace.) This entry is initially eight characters.
Justify Right Margin	Entered as yes or no. If yes, the right margin will be justified (even) in the formats that 'fill.' Two spaces will be left after sentences. If no, the right margin will be left ragged. This entry is initially yes.
Footnote Placement	Entered as inline, bottom, or endnote. This entry controls where the text of a footnote will appear. If inline, the text is enclosed in brackets and appears at the point of reference. If bottom, the text appears at the bottom of the page. If endnote, the text appears at the end of the document (exactly as if a NOTE directive was used). This entry is initially bottom.
Superscript Footnote References	Entered as yes or no. If yes, a footnote reference (for footnote placement values of bottom or end) or an endnote reference will be printed as a superscript. If no, the reference will be enclosed in brackets. This entry is initially yes.

Pad Super- and Subscripts	Entered as yes or no. If yes, extra (vertical) whitespace is left above superscripts and below subscripts to improve readability. If no, the extra space is not left. This entry is initially <b>no</b> . If the device cannot microfeed (space in small amounts vertically), this entry is forced to yes.
Four-level Sectioning (Numbering)	Entered as yes or no. If yes, the CHAPTER and APPENDIX directives are the highest level of section numbering. If no, the section and appendixsection directives are the highest. This entry is initially <b>yes</b> .
Above	Entered in micas. This entry is the amount of vertical space to leave when beginning general text or list formats. It is initially 423 micas (1 line).
Below	Entered in micas. This entry is the amount of vertical space to leave when ending a general text or list format. It is initially 423 micas (1 line).
End Space Size	Entered in bytes. Perfect Writer divides up available memory into two pieces: the page space and the end space. The end space is used to store table of contents entries, index entries, and end notes. The page space is used to store everything else. On documents that do not use any of the mentioned items, the size of the end space can be zero. This parameter allows you to vary the allocation depending upon your needs. This value is initially 1000. On a 64K system, you can increase the end space to around 8000 bytes and still have a reasonable amount of space left for the page.

While not a STYLE option proper, there is an additional item that you might want to control. This is the amount of space which Perfect Writer automatically leaves at the end of a sentence. When Perfect Writer detects a sentence end, it normally leaves an additional amount of whitespace equal to the width of the Delete (Control-?, 127 decimal) character. This will ordinarily be the width of a space. If your device has a proportional font, you can easily change this value. If your device does not have a proportional font, you can change this value by creating a width table with all of the widths the same (i.e., the fixed width for your printer) and then changing the width of the Delete character.

## **Testing Your Printer**

Now that you have told the printer configuration program about your system, you should try Perfect Writer on the small sample file, TEST.MSS, provided. Type

```
pf test
```

If this seemed to work, try printing the file:

```
pp test
```

The result should look basically like the sample at the end of this section.

If you are transferred to the main menu after typing 'pf test.mss' and all went well, then select "p" followed by the filename 'test.mss' when asked. Enter "G" from the print menu.

Sample output from TEST.MSS

**This is boldface.**

This is underscore (type u).

This is underscore (type un).

This is underscore (type ux).

*This is italic.*

This is a test of the Perfect Writer super- and subscripting commands. Here  
super

is a        and a        script. Is there enough vertical space around them? Here  
                            sub

is some more text just to be sure we can see where the left and right margins  
are and what the interline spacing is.

## **Completing the Installation**

After you have finished configuring Perfect Writer, make a third backup disk of the work you have just done. (Don't overwrite your original Perfect Writer disk!)

When all is done you should have three disks. The first one is the distribution disk. The second and third are identical and contain Perfect Writer as configured to your console. Take the original disk and one of the configured copies and put them safely away, removing the "write enable" tabs, which will ensure that you will never inadvertently overwrite them with other text. You are left with one working disk, which will be used for your day to day operations.

## **SETTING UP PERFECT WRITER**

If you have a 150K Disk then we recommend the following set-up:

**Disk B (Editing Disk)**

MENU.COM	30K
PW.COM	34
PF.COM	36
PF.DAT	6
PP.COM	<u>34</u>
	140K

**Disk A (File Disk)** This disk will contain your text files.

PW.SWP	64
--------	----

**Disk B (During Configuration)**

PWCONFIG.COM	
PFCONFIG.COM	
PF.DAT	

---

If you have a 250K disk, then we recommend the following set-up:

**Disk B (Editing Disk)**

MENU.COM	30
PW.COM	34
PF.COM	36
PP.COM	34
PF.DAT	6
PW.SWP	<u>64</u>
	214

**Disk A (Configuration Disk)**

PFCONFIG.COM	34
PWCONFIG.COM	34
PF.DAT	6

**FUNCTS.TXT File**

(C) 1982 Perfect Software, Inc. All rights reserved.

Fill Mode is the default mode.

<b>char name</b>	<b>function</b>	<b>description</b>
0 C-@	MSETMRK	Sets the mark at the point.
1 C-A	MBLINE	Point to beginning of line.
2 C-B	MPREVCHA	Backwards one character.
3 C-C	MNOTIMPL	'Unknown command'.
4 C-D	MDELCHAR	Deletes Next Character.
5 C-E	MFLINE	End of line.
6 C-F	MNEXTCHA	Forward Character.
7 C-G	MABORT	Cancel the current prefix.
8 C-H	MPREVCHA	Backwards one character.
9 <TAB>	MINsert	Inserts the character at the point.
10 <NL>	MINDREST	Indent rest of line same as this.
11 C-K	MDELLIN	Kill line.
12 C-L	MNEWDSP	Scroll-Redisplay.
13 <CR>	MNEWLIN	Carriage Return.
14 C-N	MNEXTLIN	Next line.
15 C-O	MOPENLIN	Open line.
16 C-P	MPREVLIN	Previous line.
17 C-Q	MQUOTE	Quote Character.
18 C-R	MINCRSEA	Reverse Search.
19 C-S	MINCSEAR	Forward Search.
20 C-T	MSWAPCHA	Transpose Characters.
21 C-U	MARG	Universal Argument Prefix.
22 C-V	MNEXTPAG	View Next Screen.
23 C-W	MDELRGN	Delete Region.
24 C-X	MCTRLX	eXtended Command Prefix.
25 C-Y	MYANK	Yank Kill Buffer.
26 C-Z	MPREVPAg	View Previous Screen.
27 C-[	MMETA	Meta Prefix.
28 C-\	MDELINDE	Delete indentation on current line.
29 C-]	MNOTIMPL	'Unknown command'.
30 C-†	MTOGLC	Toggle case of character at point.
31 C-	MNOTIMPL	'Unknown command'.
32 <SP>	MINsert	Inserts the character at the point.
33 !	MINsert	Inserts the character at the point.
34 "	MINsert	Inserts the character at the point.

35	#	MININSERT	Inserts the character at the point.
36	\$	MININSERT	Inserts the character at the point.
37	%	MININSERT	Inserts the character at the point.
38	&	MININSERT	Inserts the character at the point.
39	'	MININSERT	Inserts the character at the point.
40	(	MININSERT	Inserts the character at the point.
41	)	MININSERT	Inserts the character at the point.
42	*	MININSERT	Inserts the character at the point.
43	+	MININSERT	Inserts the character at the point.
44	,	MININSERT	Inserts the character at the point.
45	-	MININSERT	Inserts the character at the point.
46	.	MININSERT	Inserts the character at the point.
47	/	MININSERT	Inserts the character at the point.
48	0	MININSERT	Inserts the character at the point.
49	1	MININSERT	Inserts the character at the point.
50	2	MININSERT	Inserts the character at the point.
51	3	MININSERT	Inserts the character at the point.
52	4	MININSERT	Inserts the character at the point.
53	5	MININSERT	Inserts the character at the point.
54	6	MININSERT	Inserts the character at the point.
55	7	MININSERT	Inserts the character at the point.
56	8	MININSERT	Inserts the character at the point.
57	9	MININSERT	Inserts the character at the point.
58	:	MININSERT	Inserts the character at the point.
59	;	MININSERT	Inserts the character at the point.
60	◀	MININSERT	Inserts the character at the point.
61	=	MININSERT	Inserts the character at the point.
62	▶	MININSERT	Inserts the character at the point.
63	?	MININSERT	Inserts the character at the point.
64	@	MININSERT	Inserts the character at the point.
65	A	MININSERT	Inserts the character at the point.
66	B	MININSERT	Inserts the character at the point.
67	C	MININSERT	Inserts the character at the point.
68	D	MININSERT	Inserts the character at the point.
69	E	MININSERT	Inserts the character at the point.
70	F	MININSERT	Inserts the character at the point.
71	G	MININSERT	Inserts the character at the point.
72	H	MININSERT	Inserts the character at the point.
73	I	MININSERT	Inserts the character at the point.
74	J	MININSERT	Inserts the character at the point.

---

---

75	K	MINsert	Inserts the character at the point.
76	L	MINsert	Inserts the character at the point.
77	M	MINsert	Inserts the character at the point.
78	N	MINsert	Inserts the character at the point.
79	O	MINsert	Inserts the character at the point.
80	P	MINsert	Inserts the character at the point.
81	Q	MINsert	Inserts the character at the point.
82	R	MINsert	Inserts the character at the point.
83	S	MINsert	Inserts the character at the point.
84	T	MINsert	Inserts the character at the point.
85	U	MINsert	Inserts the character at the point.
86	V	MINsert	Inserts the character at the point.
87	W	MINsert	Inserts the character at the point.
88	X	MINsert	Inserts the character at the point.
89	Y	MINsert	Inserts the character at the point.
90	Z	MINsert	Inserts the character at the point.
91	[	MINsert	Inserts the character at the point.
92	\	MINsert	Inserts the character at the point.
93	]	MINsert	Inserts the character at the point.
94	↑	MINsert	Inserts the character at the point.
95	^	MINsert	Inserts the character at the point.
96	a	MINsert	Inserts the character at the point.
98	b	MINsert	Inserts the character at the point.
99	c	MINsert	Inserts the character at the point.
100	d	MINsert	Inserts the character at the point.
101	e	MINsert	Inserts the character at the point.
102	f	MINsert	Inserts the character at the point.
103	g	MINsert	Inserts the character at the point.
104	h	MINsert	Inserts the character at the point.
105	i	MINsert	Inserts the character at the point.
106	j	MINsert	Inserts the character at the point.
107	k	MINsert	Inserts the character at the point.
108	l	MINsert	Inserts the character at the point.
109	m	MINsert	Inserts the character at the point.
110	n	MINsert	Inserts the character at the point.
111	o	MINsert	Inserts the character at the point.
112	p	MINsert	Inserts the character at the point.
113	q	MINsert	Inserts the character at the point.
114	r	MINsert	Inserts the character at the point.

---

115	s	MINSERT	Inserts the character at the point.
116	t	MINSERT	Inserts the character at the point.
117	u	MINSERT	Inserts the character at the point.
118	v	MINSERT	Inserts the character at the point.
119	w	MINSERT	Inserts the character at the point.
120	x	MINSERT	Inserts the character at the point.
121	y	MINSERT	Inserts the character at the point.
122	z	MINSERT	Inserts the character at the point.
123	{	MINSERT	Inserts the character at the point.
124	:	MINSERT	Inserts the character at the point.
125	}	MINSERT	Inserts the character at the point.
126	~	MINSERT	Inserts the character at the point.
127	<DEL>	MRDELCHA	Delete Previous Character.
128	M-C-@	MNOTIMPL	'Unknown command'.
129	M-C-A	MNOTIMPL	'Unknown command'.
130	M-C-B	MNOTIMPL	'Unknown command'.
131	M-C-C	MNOTIMPL	'Unknown command'.
132	M-C-D	MNOTIMPL	'Unknown command'.
133	M-C-E	MNOTIMPL	'Unknown command'.
134	M-C-F	MNOTIMPL	'Unknown command'.
135	M-C-G	MABORT	Cancel the current prefix.
136	M-C-H	MNOTIMPL	'Unknown command'.
137	M-<TAB>	MNOTIMPL	'Unknown command'.
138	M-<NL>	MDROPLIN	Drop remainder of line vertically.
139	M-C-K	MDELELIN	Kill Entire Line.
140	M-C-L	MRNEWDSP	Reverse Scroll Redisplay.
141	M-<CR>	MNOTIMPL	'Unknown command'.
142	M-C-N	MNOTIMPL	'Unknown command'.
143	M-C-O	MCLOSEWH	Close up whitespace including NL.
144	M-C-P	MNOTIMPL	'Unknown command'.
145	M-C-Q	MNOTIMPL	'Unknown command'.
146	M-C-R	MQRYRPLC	Query Replace.
147	M-C-S	MNOTIMPL	'Unknown command'.
148	M-C-T	MNOTIMPL	'Unknown command'.
149	M-C-U	MNOTIMPL	'Unknown command'.
150	M-C-V	MNOTIMPL	'Unknown command'.
151	M-C-W	MMAKEDEL	Turn on + in Mode Line.
152	M-C-X	MNOTIMPL	'Unknown command'.
153	M-C-Y	MNOTIMPL	'Unknown command'.
154	M-C-Z	MNOTIMPL	'Unknown command'.

---

155	M-C-[	MNOTIMPL	'Unknown command'.
156	M-C-\	MNOTIMPL	'Unknown command'.
157	M-C-]	MNOTIMPL	'Unknown command'.
158	M-C-^	MNOTIMPL	'Unknown command'.
159	M-C-	MNOTIMPL	'Unknown command'.
160	M-<SP>	MSETMRK	Sets the mark at the point.
161	M-!	MNOTIMPL	'Unknown command'.
162	M-''	MNOTIMPL	'Unknown command'.
163	M-#	MNOTIMPL	'Unknown command'.
164	M-\$	MNOTIMPL	'Unknown command'.
165	M-%	MNOTIMPL	'Unknown command'.
166	M-&	MNOTIMPL	'Unknown command'.
167	M-'	MNOTIMPL	'Unknown command'.
168	M-{	MNOTIMPL	'Unknown command'.
169	M-}	MNOTIMPL	'Unknown command'.
170	M-*	MNOTIMPL	'Unknown command'.
171	M-+	MNOTIMPL	'Unknown command'.
172	M-,	MNOTIMPL	'Unknown command'.
173	M--	MNOTIMPL	'Unknown command'.
174	M-.	MNOTIMPL	'Unknown command'.
175	M-/	MNOTIMPL	'Unknown command'.
176	M-0	MARGDGT	Numeric Digit Argument Prefix.
177	M-1	MARGDGT	Numeric Digit Argument Prefix.
178	M-2	MARGDGT	Numeric Digit Argument Prefix.
179	M-3	MARGDGT	Numeric Digit Argument Prefix.
180	M-4	MARGDGT	Numeric Digit Argument Prefix.
181	M-5	MARGDGT	Numeric Digit Argument Prefix.
182	M-6	MARGDGT	Numeric Digit Argument Prefix.
183	M-7	MARGDGT	Numeric Digit Argument Prefix.
184	M-8	MARGDGT	Numeric Digit Argument Prefix.
185	M-9	MARGDGT	Numeric Digit Argument Prefix.
186	M-:	MNOTIMPL	'Unknown command'.
187	M-;	MNOTIMPL	'Unknown command'.
188	M-<	MTOSTART	Beginning of Buffer.
189	M-=	MNOTIMPL	'Unknown command'.
190	M->	MTOEND	End of buffer.
191	M-?	MNOTIMPL	'Unknown command'.
192	M-@	MNOTIMPL	'Unknown command'.
193	M-A	MBSENT	Beginning of sentence.
194	M-B	BWORD	Backwards word.

---

195	M-C	MCAPWORD	Capitalize word.
196	M-D	MDELWORD	Kill Forward Word.
197	M-E	MFSENT	Forward Sentence.
198	M-F	FWORD	Forward Word.
199	M-G	MNOTIMPL	'Unknown command'.
200	M-H	MMRKPARA	Mark Whole Paragraph.
201	M-I	MNOTIMPL	'Unknown command'.
202	M-J	MINDNL	Indent Subsequent New Line Same As This.
203	M-K	MDELENT	Kill Sentence.
204	M-L	MLOWWORD	Lowercase Word.
205	M-M	MNOTIMPL	'Unknown command'.
206	M-N	MFPARA	Forward Paragraph.
207	M-O	MOPENIND	Indent Leading New Line Same As This.
208	M-P	MBPARA	Back Paragraph.
209	M-Q	MFILLPAR	Fill Paragraph.
210	M-R	MREPLACE	Global Replace.
211	M-S	MCNTRLIN	Center Line.
212	M-T	MSWAPWOR	Transpose Word.
213	M-U	MUPWORD	Uppercase Word.
214	M-V	MNOTIMPL	'Unknown command'.
215	M-W	MCOPYRGN	Copy Region to Kill Buffer.
216	M-X	MNOTIMPL	'Unknown command'.
217	M-Y	MNOTIMPL	'Unknown command'.
218	M-Z	MNOTIMPL	'Unknown command'.
219	M-[	MNOTIMPL	'Unknown command'.
220	M-	MDELWHIT	Delete Whitespace.
221	M-]	MNOTIMPL	'Unknown command'.
222	M-↑	MNOTIMPL	'Unknown command'.
223	M-	MNOTIMPL	'Unknown command'.
224	M-`	MNOTIMPL	'Unknown command'.
225	M-a	MBSENT	Beginning of sentence.
226	M-b	BWORD	Backwards word.
227	M-c	MCAPWORD	Capitalize Word.
228	M-d	MDELWORD	Kill Forward Word.
229	M-e	MFSENT	Forward Sentence.
230	M-f	FWORD	Forward Word.
231	M-g	MNOTIMPL	'Unknown command'.
232	M-h	MMRKPARA	Mark Whole Paragraph.

---

233	M-i	MNOTIMPL	'Unknown command'.
234	M-j	MINDNL	Indent Subsequent New Line Same As This.
235	M-k	MDELSENT	Kill Sentence.
236	M-l	MLOWWORD	Lowercase Word.
237	M-m	MNOTIMPL	'Unknown command'.
238	M-n	MFPARA	Forward Paragraph.
239	M-o	MOPENIND	Indent Leading New Line Same As This.
240	M-p	MBPARA	Back Paragraph.
241	M-q	MFILLPAR	Fill Paragraph.
242	M-r	MREPLACE	Global Replace.
243	M-s	MCNTRLIN	Center Line.
244	M-t	MSWAPWOR	Transpose Word.
245	M-u	MUPWORD	Uppercase Word.
246	M-v	MNOTIMPL	'Unknown command'.
247	M-w	MCOPYRGN	Copy Region to Kill Buffer.
248	M-x	MNOTIMPL	'Unknown command'.
249	M-y	MNOTIMPL	'Unknown command'.
250	M-z	MNOTIMPL	'Unknown command'.
251	M-{	MNOTIMPL	'Unknown command'.
252	M:-	MNOTIMPL	'Unknown command'.
253	M-}	MNOTIMPL	'Unknown command'.
254	M-~	MNOTIMPL	'Unknown command'.
255	M-<DEL>	MRDELWOR	Kill Previous Word.
256	X-C-@	MNOTIMPL	'Unknown command'.
257	X-C-A	MNOTIMPL	'Unknown command'.
258	X-C-B	MLSTBUFF	List Buffers.
259	X-C-C	MEXIT	Exit Perfect Writer.
260	X-C-D	MNOTIMPL	'Unknown command'.
261	X-C-E	MNOTIMPL	'Unknown command'.
262	X-C-F	MFINDFIL	Find File.
263	X-C-G	MABORT	Cancel the current prefix.
264	X-C-H	MNOTIMPL	'Unknown command'.
265	X-<TAB>	MSETTABS	Set Tab Stops.
266	X-<NL>	MNOTIMPL	'Unknown command'.
267	X-C-K	MNOTIMPL	'Unknown command'.
268	X-C-L	MNOTIMPL	'Unknown command'.
269	X-<CR>	MDELMODE	Delete Mode from Mode List.

270	X-C-N	MNOTIMPL	'Unknown command'.
271	X-C-O	MNOTIMPL	'Unknown command'.
272	X-C-P	MNOTIMPL	'Unknown command'.
273	X-C-Q	MNOTIMPL	'Unknown command'.
274	X-C-R	MFILEREA	Read File.
275	X-C-S	MFILESAV	Save File.
276	X-C-T	MNOTIMPL	'Unknown command'.
277	X-C-U	MNOTIMPL	'Unknown command'.
278	X-C-V	MNXTOTHR	View other window's next screen.
279	X-C-W	MFILEWRI	Write File.
280	X-C-X	MSWAPMRK	Swap Point and Mark.
281	X-C-Y	MNOTIMPL	'Unknown command'.
282	X-C-Z	MPRVOTHR	View other window's previous screen.
283	X-C-[	MNOTIMPL	'Unknown command'.
284	X-C-\	MNOTIMPL	'Unknown command'.
285	X-C-]	MNOTIMPL	'Unknown command'.
286	X-C↑	MNOTIMPL	'Unknown command'.
287	X-C-	MNOTIMPL	'Unknown command'.
288	X-<SP>	MNOTIMPL	'Unknown command'.
289	X-!	MNOTIMPL	'Unknown command'.
290	X-"	MNOTIMPL	'Unknown command'.
291	X-#	MNOTIMPL	'Unknown command'.
292	X-\$	MNOTIMPL	'Unknown command'.
293	X-%	MNOTIMPL	'Unknown command'.
294	X-&	MNOTIMPL	'Unknown command'.
295	X-'	MNOTIMPL	'Unknown command'.
296	X-(	MNOTIMPL	'Unknown command'.
297	X-)	MNOTIMPL	'Unknown command'.
298	X-*	MWHATVER	Echos the version and release date.
299	X-+	MNOTIMPL	'Unknown command'.
300	X-,	MNOTIMPL	'Unknown command'.
301	X--	MNOTIMPL	'Unknown command'.
302	X-.	MSETINDE	Set Indent Column.
303	X-/	MNOTIMPL	'Unknown command'.
304	X-0	MNOTIMPL	'Unknown command'.
305	X-1	MONEWIND	One Window Mode.
306	X-2	MTWOWIND	Two Window Mode.
307	X-3	MNOTIMPL	'Unknown command'.
308	X-4	MNOTIMPL	'Unknown command'.
309	X-5	MNOTIMPL	'Unknown command'.

---

310	X-6	MNOTIMPL	'Unknown command'.
311	X-7	MNOTIMPL	'Unknown command'.
312	X-8	MNOTIMPL	'Unknown command'.
313	X-9	MNOTIMPL	'Unknown command'.
314	X-:	MNOTIMPL	'Unknown command'.
315	X-;	MNOTIMPL	'Unknown command'.
316	X-<	MUNDNTRG	Untent Region.
317	X-=	MPRINTPO	Print Current Position.
318	X->	MINDNTRG	Indent Region.
319	X-?	MNOTIMPL	'Unknown command'.
320	X-@	MNOTIMPL	'Unknown command'.
321	X-A	MNOTIMPL	'Unknown command'.
322	X-B	MSWITCHT	Switch to another buffer.
323	X-C	MNOTIMPL	'Unknown command'.
324	X-D	MNOTIMPL	'Unknown command'.
325	X-E	MNOTIMPL	'Unknown command'.
326	X-F	MSETFILL	Set Fill Column.
327	X-G	MNOTIMPL	'Unknown command'.
328	X-H	MNOTIMPL	'Unknown command'.
329	X-I	MINSSFI	Insert File.
330	X-J	MNOTIMPL	'Unknown command'.
331	X-K	MKILLBUF	Delete Buffer.
332	X-L	MNOTIMPL	'Unknown command'.
333	X-M	MADDMODE	Add Mode To Mode List.
334	X-N	MNOTIMPL	'Unknown command'.
335	X-O	MSWPWIND	Switch to other window.
336	X-P	MNOTIMPL	'Unknown command'.
337	X-Q	MNOTIMPL	'Unknown command'.
338	X-R	MWRTRGN	Write Region to file.
339	X-S	MGOSPELL	Call the spelling checker from PW.
340	X-T	MNOTIMPL	'Unknown command'.
341	X-U	MNOTIMPL	'Unknown command'.
342	X-V	MNOTIMPL	'Unknown command'.
343	X-W	MNOTIMPL	'Unknown command'.
344	X-X	MNOTIMPL	'Unknown command'.
345	X-Y	MNOTIMPL	'Unknown command'.
346	X-Z	MNOTIMPL	'Unknown command'.
347	X-[	MNOTIMPL	'Unknown command'.
348	X-\`	MTRIMWHI	Globally delete trailing whitespace.
349	X-]	MNOTIMPL	'Unknown command'.

---

350	X-†	MGROWWIN	Grow current window.
351	X-	MNOTIMPL	'Unknown command'.
352	X-`	MNOTIMPL	'Unknown command'.
353	X-a	MNOTIMPL	'Unknown command'.
354	X-b	MSWITCHT	Switch to another buffer.
355	X-c	MNOTIMPL	'Unknown command'.
356	X-d	MNOTIMPL	'Unknown command'.
357	X-e	MNOTIMPL	'Unknown command'.
358	X-f	MSETFILL	Set Fill Column.
359	X-g	MNOTIMPL	'Unknown command'.
360	X-h	MNOTIMPL	'Unknown command'.
361	X-i	MINSSFI	Insert File.
362	X-j	MNOTIMPL	'Unknown command'.
363	X-k	MKILLBUF	Delete Buffer.
364	X-l	MNOTIMPL	'Unknown command'.
365	X-m	MADDMODE	Add Mode to Mode List.
366	X-n	MNOTIMPL	'Unknown command'.
367	X-o	MSWPWIND	Switch to other window.
368	X-p	MNOTIMPL	'Unknown command'.
369	X-q	MNOTIMPL	'Unknown command'.
370	X-r	MWRTGRN	Writer Region to file.
371	X-s	MGOSPELL	Call the spelling checker from PW.
372	X-t	MNOTIMPL	'Unknown command'.
373	X-u	MNOTIMPL	'Unknown command'.
374	X-v	MNOTIMPL	'Unknown command'.
375	X-w	MNOTIMPL	'Unknown command'.
376	X-x	MNOTIMPL	'Unknown command'.
377	X-y	MNOTIMPL	'Unknown command'.
378	X-z	MNOTIMPL	'Unknown command'.
379	X-{	MNOTIMPL	'Unknown command'.
380	X-:	MNOTIMPL	'Unknown command'.
381	X-}	MNOTIMPL	'Unknown command'.
382	X-~	MNOTIMPL	'Unknown command'.
383	X-<DEL>	MNOTIMPL	'Unknown command'.

## **Appendix B FORMAT ERROR MESSAGES**

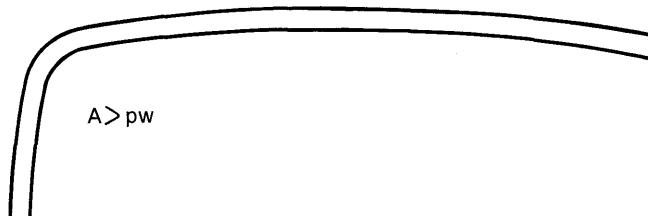
This is a listing of error messages that Perfect Formatter can generate. Each message is given with an explanation of what might have gone wrong, together with some suggestion about correcting the error. Single quotes, ' ', are used to specify the particular text that is producing the problem.

Error messages fall into three classes: "ordinary," "internal," and "system." In the following discussion, the error messages are listed alphabetically within each category.

To help you find the error in your document, Perfect Formatter will, whenever possible, indicate the line number where the error occurred.

## Finding the Error

To find the error that Perfect Writer has indicated:



1. Re-enter Perfect Writer's screen editor using the Begin Procedure described in Chapter III, page 1.
2. Call up the file containing the error, using the FIND FILE command: (See Chapter IX, page 4.)



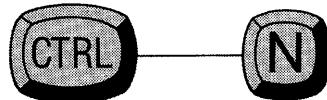
3. Position the cursor at the beginning of the file document.

4. Advance the cursor to the line which holds the error using the UNIVERSAL REPEAT command:



followed by the line number,

followed by the NEXT LINE command:



Perfect Writer will advance the cursor to **one line beyond** the line containing the error (Remember, the search began on the **first line**.)

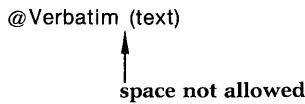
---

## **Ordinary Error Messages**

Ordinary error messages result from some explicit error in the text, usually an erroneous or missing command statement. These are errors that you should be able to correct by reviewing the commands that you have given.

### **Bad opening character 'char'**

Indicates that the first character following the keyword of a format command was not a 'fence.' Either the fence is missing altogether, or a space exists between the command and the fence. **Example:**



### **Can't open file 'filename'**

Perfect Writer cannot find the file that you wish to format and print. Several possible causes exist: you have mistyped the filename, the file is not on the disk you have specified, or the file does not exist.

### **@CHAPTER and @APPENDIX have been disabled**

You have indicated to Perfect Writer through the 'chapters' style parameter that chapters and appendices will not be used. Yet, one or more CHAPTER and APPENDIX commands have been encountered in the document.

### **Exiting but in a 'name' environment**

A format option has not been completely 'fenced.' Either a closing 'fence' is missing from some format option, or the beginning of an extra and unneeded format option has been inadvertently included somewhere in the text.

### Extra arguments to 'command'

You have specified too many 'arguments' for a particular command.  
For example:

@Blankspace(2 lines, 6 lines)

Perfect Writer does not know which argument to use.

The @BLANKSPACE command only takes one argument, in this case '2 lines' or '6 lines,' but not both.

### File read error

The file that you are attempting to format and print is somehow 'messed up' or not what it should be. Perfect Writer stops attempting to format it, and proceeds to the next file to be formatted and printed.

Possible causes are that you have changed disks and not alerted Perfect Writer to the file allocation of the new disk, by performing a 'Control-C' after inserting the new disk. Or perhaps the file you are attempting to format and print was prepared on another word processor which Perfect Writer is not familiar with.

### Incorrect Environment Closing. Tried to end a 'name' environment. A 'name' environment is open, and it should be closed with 'closer'

In a BEGIN/END format command form, the END statement did not in some way match the BEGIN statement. For example,

@BEGIN(QUOTATION)  
.....  
.....  
.....  
@END(VERSE)

do not match

### **Invalid number 'rest of string'**

In some command an alphabetical character was used in place of a number. For example,

@BLANKSPACE(1o lines)  
The small letter 'o' instead of the number zero '0'

### **Missing argument to 'command'**

Similar to 'Bad opening character.' The first character following the keyword of a command is not a 'fence.' Either that or a space has been mistakenly inserted between the keyword and the fence.

### **Missing dimension.**

You have not supplied the complete form of a dimensioned number. For example,

@BLANKSPACE(5)	incorrect
@BLANKSPACE(5 lines)	correct

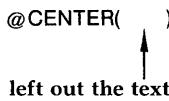
### **Missing numeric argument**

You have not supplied a necessary number to a command. For example,

@BLANKSPACE(lines)	incorrect
@BLANKSPACE(5 lines)	correct

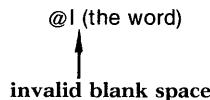
### **Missing 'single' argument**

You have not supplied an argument to a command that takes a **single** argument. For example,



```
@CENTER( )  
↑  
left out the text
```

Or an extra space exists between the command and the 'fence.' For example,



```
@I (the word)  
↑  
invalid blank space
```

### **Non-numeric value 'name'**

Some commands, such as the @REF command (see Chapter XVII, page 16) have tried to obtain a **numeric** value from a variable that does not represent a numeric value. (See the discussion on variables, Chapter XVII, page 11.)

### **Output file write error**

Perfect Writer was unable to store the formulated version of your document on disk.

Possible causes are that no more room on the disk exists for it, or that you have changed disks and have not informed Perfect Writer of the new disk's storage allocation patterns (using a Control-C after inserting the new disk).

---

**@REF: Variable not found 'name'**

Using the @REF command, you are attempting to reference a variable that does not exist.

Either you have forgotten to define the variable (using the @SET commands) or you are mistyping the name of the variable in your reference.

**@REF: 'name' does not contain a number**

Using the @REF command, you are attempting to reference a variable that contains not a numeric value (which @REF requires), but an alphabetic string. You have possibly defined the variable using the @STRING command rather than the @SET command. (See discussion of variables, Chapter XVII, page 11.)

**@SET requires two arguments**

The @SET command requires two arguments to define a variable: 1) the variable name and 2) the value which the name will represent. You have supplied only one of these. Example:

@SET(page)                  incorrect

@SET(page = 5)              correct

### @STRING requires two arguments

The @STRING command requires two ingredients to define a variable: 1) the variable name and 2) the value which the name will represent. You have supplied only one of these. Example:

@STRING(school)	incorrect
@STRING(school = Harvard University)	correct

### @STRING: Variable not found 'name'

In attempting to define a variable in terms of another **second** variable, Perfect Writer could not locate the second variable. Example:

@STRING(school = university)  
variable "university" not defined

You neglected to define university as a variable [@STRING(university= )].

### @TITLE: Unknown option 'name'

In attempting to reference a chapter or other section title, using the @TITLE command, you have inadvertently used the wrong variable. TITLE takes only the following variable forms: 'chapter,' 'section,' etc., **not** CHAPERTITLE, SECTIONTITLE, etc. (See Chapter XVII, page 15.)

### Undefined or non-string selector variable for @CASE 'name'

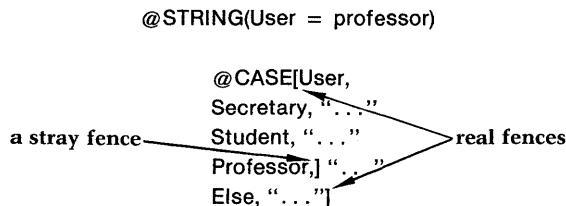
The selector variable you have used with the @CASE format option was either not defined, or it was defined as having a numeric value.

Possible causes are mistyping the variable name, or not using the @STRING command to properly define the variable. (See discussion on the @CASE format option, Chapter XVII, pages 1-3.)

### **Unexpected end of @CASE argument**

An alternative text for the @CASE format option was successfully selected but the text that it presented was **missing**.

A possible cause is an extra or stray 'fence' that prematurely closed the @CASE format. For example:



### **Unknown argument to @PAGEHEADING or @PAGEFOOTING 'argument'**

You have improperly designated the arguments for a PAGEHEADING or PAGEFOOTING command, which can only be 'left, center, right, line, even, or odd.' Either the argument was mistyped or an internal pagefooting delimiter was not matched properly. (See PAGEFOOTING command, Chapter XVI, pages 12-15.)

### **Unknown argument to @STYLE 'argument'**

You have provided an unknown argument in a @STYLE parameter command. Possibly you have mistyped the argument. Example:



### **Unknown command 'name'**

Perfect Writer cannot identify the command. Mistyped? (See the Reference Card for the proper form of all commands.)

### **Unknown environment 'name'**

The format option specified in a BEGIN-END command form was either unknown or not a format option that could be used with this command form. For example:

`@BEGIN(INCLUDE)`

cannot use 'Include'  
in this way

or

`@BEGIN(ENUMERATE)`

misspelled

### **Unknown unit of measure 'unit'**

Perfect Writer could not identify the unit of measure supplied in a command. For example:

`@BLANKSPACE(6 miles)`

### **Unknown value for Note clause of @STYLE 'value'**

The value supplied in the 'notes' style parameter was not 'endnote,' 'bottom,' or 'inline,' but something that Perfect Writer could not identify. Has it been mistyped?

### **@VALUE: Variable not found 'name'**

Using the @VALUE command you have referenced a variable that does not exist or which has not been defined using a @STRING command. (See discussion of variables, Chapter XVII, pages 8-13.)

---

**Warning: Argument terminated by paragraph break**

A command which takes a single argument (such as the italicize command '@I') has been terminated by a paragraph break rather than its closing delimiter (which was probably missing). This is a safety feature to prevent 'runaway' commands. Example:

This is an example, nothing more. @I(This  
sentence is italicized. ← no closing fence here so  
This is an example, nothing more. the italicizing was stopped  
by a paragraph break.

**Warning: Footnote wrap not handled properly**

A footnote reference has occurred too near the bottom of a page, which has resulted in the footnote (which itself will go at the bottom) being partially or completely lost.

To correct this, try placing a @NEWPAGE command before the footnote, which will cause Perfect Writer to skip to a new page before encountering the footnote. On the new page it will be able to save space for the footnote.

**'Yes' or 'No' argument required; 'argument' was given**

Something other than a 'yes' or 'no' was supplied to a command requiring such an answer.

## **Internal Error Messages**

Internal error messages indicate usually that some internal program limit has been exceeded, that storage space has been used up, that a line is too long, that a buffer has overflowed. There was nothing wrong with the input document **per se**, although changing the document (such as breaking it into smaller segments) may help to correct the problem.

### **Current token overflow**

A single string of alphanumeric characters, spaces, or tabs (more than 100 in a row) was encountered in the text. This single 'word' was too long for Perfect Writer to handle. In such a situation, Perfect Writer discards the last 50 characters of the string and continues.

### **Input buffer overflow**

Certain commands such as @FOOT, @NOTE, and the numbered section headings require Perfect Writer to save text for later inclusion in a separate table of contents or notes section.

This error means that the segment of text being saved for that purpose exceeded the maximum of 300 characters. Also, a closing 'fence' could simply have been omitted from a command, making the text to be footnoted or placed in a table of contents text much longer than you intended.

### **Out of memory**

When formatting, Perfect Writer divides its machine's memory into two parts: that which holds the running text, called 'page space,' and that which accumulates footnotes, table of contents, and index entries, called 'end space.'

An error occurs should any of these become full.

---

### **Put buffer overflow**

A single string of more than 200 characters was given to be buffered for formatting. This is only a warning. Perfect Writer throws away the last 50 characters and continues.

### **Too many lines**

Perfect Writer gathers and stores up to 100 lines at a time for printing on any single page. If a page length has been specified that exceeds the 100 line capacity (that is, your paper is extremely long), this error will occur. However, it is only a **warning**. Perfect Writer begins a new page and continues printing (leaving the unused space blank).

### **Too many tokens for a single line**

This error occurs if a line in a format which does not 'wrap' long lines contains more than 50 'tokens'—i.e. 'words' or 'character units.' However, this is only a warning; Perfect Writer discards the last 20 tokens and continues.

Possible causes are not having newlines in non-wrapped paragraphs (an artifact of some text editors) or using lots of very short words.

## **Execution Error Messages**

Execution error messages concern errors that occur before actual formatting of a file begins.

### **Can't find the configuration file, PF.DAT**

Perfect Writer cannot find the PF.DAT file which holds the specifications for your printing device, which you supplied during Printer Configuration. You may have to transfer a copy of PF.DAT from your backup configured disk.

### **Default or console device not defined**

No default or console device has been defined for Perfect Writer in the configuration file.

### **Device 'name' not found, using default**

Perfect Writer could not find the printing device you indicated. It is using the default device instead.

### **Fatal: Unable to read message from configuration file**

Data specifications for formatting could not be read from the PF.DAT file. Either replace the present PF.DAT file or run the PFCONFIG program again.

## Notes

## Appendix C "SWAPPING . . . "

### **Introduction**

From time to time you will see displayed at the right hand bottom of your screen the message "swapping . . ." This message is associated with Perfect Writer's 'virtual memory architecture,' the feature that allows instantaneous display of any portion of the document you are editing. The message means that Perfect Writer is 'swapping' parts of your document which it thinks you will not be using for a while. You will know that Perfect Writer is swapping not only by the message, but a 'click' and 'whirr' at the disk drives of your computer.

Swapping is a completely automatic process that you need never concern yourself with. Usually Perfect Writer takes the opportunity to do it when you are not working at the keyboard. If it does need to swap while you are working, the delay to you will usually be no longer than two seconds.

### **"Swap File Full"**

If your document is long (65 pages or more), you may get the message "swap file full," which means that Perfect Writer has no more space in its special swap file to store the portions of your document you are not currently editing. There are several ways of dealing with this problem:

1. Delete one or more existing buffers, if you have several active, since each active buffer is using a portion of swap file space (see Chapter IX, page 9).
2. Divide your document into smaller segments and store these on disk under different file names, working afterwards on each segment individually.
3. Temporarily quit using Perfect Writer and increase the size of the swap file using the configuration program (which sets the size of the swap file initially at 64K bytes—enough for about 60 pages of text). Remember to save your buffer files before quitting or they will be lost.

## Notes

## **GLOSSARY**

back space	The backspace key is defined initially as non-destructive, thus it moves the cursor back one space without changing the text. If your keyboard does not have a delete key, you may want to swap the delete key code for the backspace key code (i.e., use the swap key option to swap hex 8 for hex 7F). This would make the backspace key destructive. Thus, when you move back you would be deleting.
break	A "break" occurs when filling is temporarily turned off. This occurs, for example, at the end of a paragraph in order to start a new line. There are specific types of breaks. <ul style="list-style-type: none"><li>• A line break in the document occurs whenever a newline character is encountered. A newline character is usually a carriage return, a line feed, or some combination of these.</li><li>• A line break in the result occurs whenever Perfect Writer places a newline character in the result.</li><li>• A paragraph break occurs in the document whenever Perfect Writer encounters a blank line.</li><li>• A page break occurs when one page of the result is filled up and a new one begun.</li></ul>
buffer	The space in the computer's memory where text is temporarily stored while the computer is on. With the virtual memory architecture used by Perfect Writer, the buffer serves as the space where part of a text file is stored while it is being edited. The buffer is the space between the computer's main (or core) memory and the disk file. It is possible to have as many as seven buffers open during an editing session.
buffer name	The name assigned to the buffer space. The buffer name consists of the first part of the filename assigned to the disk file of a document.

CP/M	This is the Control Program Monitor that provides a standard operating system for microcomputers. It acts as the interface between the computer hardware and software. The major advantage of CP/M is that it permits software to be interchangeable between computers. In addition, CP/M provides numerous operating utilities to copy disks, rename and erase files, list the contents of disks, and more.
CR	This is the symbol used to refer to the carriage return key (labeled RETURN, CR, or ENTER on the keyboard).
command	You control Perfect Writer by giving it commands. Commands can be used for either editing or formatting. Editing commands are issued by using the Control and Escape keys. Format commands begin with the at-sign ('@').
console	The "console" is the device you use to interact with Perfect Writer. Perfect Writer also prints status and error messages on the console.
CTRL	This is the prefix used in the manual to refer to Control commands. These are formed by holding down the key marked "Control" or "CTRL" on the keyboard while typing another character.
cursor	The screen display always has the terminal's blinking pointer or cursor on it. Where the cursor is on the screen is the position where Perfect Writer editing commands will affect the text. Many Perfect Writer editing commands do nothing but move the cursor to the desired position on the screen in order to perform editing operations.
DEL	The symbol used to refer to the delete key (labeled DELETE, DEL, or RUBOUT on the keyboard). This key sends an ASCII Control-?, decimal 127, or hex 7F.

device	Perfect Writer uses a format program called Perfect Formatter. Perfect Formatter always formats its results for a particular output device (terminal, line printer, etc.). For Perfect Formatter a device consists of a description of a printer and the size of the paper that is being used. Devices are defined for Perfect Formatter by using the Perfect Formatter configuration program (PFCONFIG.COM). Results from the Perfect Formatter configuration program are stored in the PF.DAT file.
disk file	Text files are stored on disk in a disk file. After editing a file on Perfect Writer it is normally stored on a disk. This provides permanent storage for the document.
document	Perfect Writer usually gets the input text from a disk file. This file will be referred to as the "document file" or just "document." The name of the file will usually end in ".MSS" (for ManuScript Source).
Echo Line	The line in the lower left corner of the screen display beneath the Mode Line is the Echo Line and is used by Perfect Writer for three purposes: <ul style="list-style-type: none"><li>• This line will echo any prefix characters (the first keystroke of any of the two-keystroke editing commands) typed. For example, if the Escape key is typed, the phrase "Meta:" will appear in the Echo Line so that you know that you have indeed typed the prefix character.</li><li>• The Echo Line is used for reading and displaying the information needed for commands which require user input (i.e., the search, storing and buffer switching commands).</li><li>• Error messages are displayed in the Echo Line and, if your terminal has a bell, it will ring to alert you to the error condition.</li></ul>
environment	An environment is a description of how the result should look. Environments control such things as the margins, the line spacing, whether lines wrap, and how paragraphs should start.

---

filling	The process of making a short line as long as possible by taking words from following lines is called "filling."
format commands	Format commands allow for the printing of text in a set manner. Format commands provide default tab settings, margin settings, line spacing, indent settings and so on. Perfect Writer offers a variety of format commands for such needs as quotations, sectioning a document, printing footnotes, justifying text, and more. Format commands always begin with the at-sign ("@") followed by the name of the format command.
justifying	"Justifying" text means taking a filled line and inserting extra space in order to make the right margin even.
left justified	"Left justified" text refers to text whose left edge is even but whose right margin is ragged.
mark	This is an invisible indicator in the text buffer. It may be set at a particular position by using the MARK SET command.
Meta	This is the name used to refer to Escape key commands referred to as "Meta commands." The commands are entered by typing the key marked Escape and then typing another character.
microfeed	A device that can "microfeed" can move down more finely than one line. Typical microfeeding devices can move down by 1/48th inch (1/8th of a line).
microspace	A device that can "microspace" can position characters more finely than the size of a character. A device that can space by half-, a quarter-, sixth, or tenth-character can microspace.

Mode Line	<p>The line just below the text window is the Mode Line. It provides important information on the current status of Perfect Writer:</p> <ul style="list-style-type: none"><li>• The user is operating in Perfect Writer rather than the operating system.</li><li>• The mode the user is currently in. There are several modes available, such as SAVE, VIEW, OVERWRITE, NORMAL, FILL, and SPELL.</li><li>• The name of the buffer your editing text is in. This name is used when switching from buffer to buffer.</li><li>• The name of the text file you are editing. If the file was called from disk, then information used to access the file will be displayed.</li><li>• The point in the text where the cursor is located in proportion to the full document (displayed as a percentage). For instance, if the cursor is in the exact middle of a document, the Mode Line will display "50%".</li><li>• In addition, the Mode Line will display a "*" if the current file has been modified but the modifications have not yet been saved.</li></ul>
move	<p>This means "move the cursor." All Perfect Writer editing operations are done by "moving to" the appropriate place in a document (and on the screen) and making necessary changes.</p>
paragraph	<p>A paragraph, as defined for the paragraph movement or filling commands goes until either a blank line, a line beginning with a tab, or a line beginning with an at-sign ("@") is reached.</p>
proportional spacing	<p>"Proportional spacing" means that different characters have different widths, for example, an 'i' is narrower than an 'm.' A device must be able to microspace in order for Perfect Writer to use a proportional type font on it.</p>
result	<p>The file where Perfect Writer puts its formatted output is called the "result" file. The name of this file will usually end in the extension ".FIN" (for FINal or FINished).</p>

right justified	"Right justified" text refers to text whose right edge is even. "Right justified" often means the same thing as "justified."
sentence	A sentence, as defined for the sentence movement and deletion commands, begins at the first word found and extends until a closing punctuation mark is found.
split screen	Allows the screen to be divided for the display of multiple file buffers. Perfect Writer allows for the screen window to be split into two windows, each displaying a different text buffer (use the TWO WINDOW command).
style parameters	These parameters permit the user to set the numerous STYLE options for the format commands. All format commands are provided with default style parameter settings.
swapping	Perfect Writer makes use of a virtual memory software architecture that allows it to edit a document much larger than the computer's core memory. This is accomplished by swapping or transferring in pages of a document from a 'swap file' as they are needed for editing. To the user this swapping is transparent because it is all done automatically. However, from time to time Perfect Writer will display the message "swapping" while it is conducting this swapping operation.
virtual memory	Allows your computer to process files much larger than its internal or core memory by creating 'virtual memory' on disk which is accessed when needed. This feature allows Perfect Writer to edit documents several times larger than your computer's core memory by creating a large swap file which is used to create 'virtual memory' on disk.
word	A word, as defined for the word movement and deletion commands, begins at the first alphanumeric character found and extends until a blank space or punctuation mark is found.
wrapping	The process of moving the text at the end of a long line onto the following line is called "wrapping."

## INDEX

### **A**

ADD MODE command VIII-2  
ADJUST TEXT command V-11  
ADDRESS format XIV-8  
alphanumeric variables XVII-11  
APPENDIX format XVI-5, A-29, B-4  
APPENDIX headings XVI-8  
at sign @ XIII-3, XIV-5

### **B**

backspace key III-4, A-11  
backup A-1, A-33  
BACKWARD CHARACTER command IV-1, IV-14  
backward references XVII-16  
BACKWARD WORD command IV-2, IV-14  
BEGIN-END command XIII-6, XV-1, B-5  
BEGINNING OF DOCUMENT command IV-10, IV-14  
BEGINNING OF LINE command IV-3, IV-14  
BEGINNING OF PARAGRAPH command IV-5, IV-14  
beginning Perfect Writer III-1  
BEGINNING OF SENTENCE command IV-4, IV-14  
below style parameter XVIII-3, A-29  
bidirectional printing A-21  
BIOS A-8, A-9  
    see CP/M  
BLANKPAGE command XVII-8  
BLANKSPACE command XVII-9, B-5  
boldface A-13  
BOLDFACE format XV-1, A-13  
BOLDFACE ITALICS format XV-3  
Bottommargin style parameter XVIII-3, A-27  
boundary mark V-7  
buffer IX-1  
buffer commands IX-4, X-2  
BUFFER DIRECTORY command IX-8, IX-10  
buffer names IX-3

---

**C**

Cancel

    GO BACK command VII-9

carriage return III-4

CASE command XVII-1, B-9, B-10

    else option XVII-3

    null option XVII-3

CENTER format XIV-7, B-7

CENTER TEXT command IV-13

Centronics printer A-22

changing screens IV-7

CHAPTER format XVI-4, B-4

Chapter style parameter XVIII-3, A-29

CLOSE INSERT command V-11, V-13

CLOSING format XIV-8

command keys II-4

command messages II-8

COMMENT command XVII-7

configuration file A-1, A-13, C-1

    PF.DAT A-13, A-17, A-25, B-15

console IX-2, A-1, A-2, A-12, A-22, B-15

    see screen

console device A-22

CONTINUE SAVING command XI-8, XI-9

Control key II-5

COPY REGION command XI-7

copying text XI-1

counters XVII-16

cursor II-2, IV-1

---

**D**

decimal A-5, A-14  
default filename VI-2  
delay count A-11  
DELETE BUFFER command IX-9  
Delete key III-4, V-1  
DELETE ENTIRE LINE command V-4, V-13, XI-2, XI-4  
DELETE LINE command V-3, V-13, XI-2, XI-4  
DELETE MODE command VIII-3  
DELETE NEXT CHARACTER command V-2, V-13  
DELETE NEXT WORD command V-2, V-13, XI-2, XI-14  
DELETE PREVIOUS CHARACTER command V-1, V-13  
DELETE PREVIOUS WORD command V-1, V-13, XI-2, XI-4  
DELETE SENTENCE FORWARD command V-5, V-13  
Deleting V-1  
DESCRIPTION format XIV-20  
DEVICE command XVIII-1, A-7  
Diablo A-21  
dimensioned numbers XVIII-2  
DIR XVIII-1  
disk storage IX-2, X-2  
DISPLAY format XIV-26  
document design commands XVIII-1

---

**E**

Echo Line II-1, II-3, V-8  
editing buffer IX-1, X-2  
END OF DOCUMENT command IV-11, IV-14  
END OF LINE command IV-3, IV-14  
END OF PARAGRAPH command IV-5, IV-14  
END OF SENTENCE command IV-4, IV-14  
end space A-29, B-13  
ENLARGE WINDOW command X-7  
Enter key III-4  
ENUMERATE format XIV-16  
environment format command XIV-1  
Epson MX-80 A-20  
error messages B-1, B-2, B-4, B-13  
Escape key II-4, II-6  
Escape-Control commands II-7  
EXAMPLE format XIV-3  
EXCHANGE CURSOR AND MARK command IV-12, IV-14

**F**

fences XIII-3, XVII-3, B-4, B-6, B-13  
file commands VI-5  
file names VI-2  
file retrieval VI-8  
Fill Mode II-3, VIII-1  
filling A-3  
.FIN file extension VII-5 to VII-7, A-8, A-15  
FIND FILE command IX-4, XI-5, B-2  
FLUSHLEFT format XIV-10  
FLUSHRIGHT format XIV-10  
Footerspacing style parameter XVIII-3, A-27  
FOOTNOTE command XVI-10, A-28, B-12, B-13  
Footpush style parameter XVI-10, XVI-11, XVIII-3  
form letters XVII-1  
format commands XIII-3, XIII-9  
formatting VII-10, XIII-9  
FORWARD CHARACTER command IV-1, IV-14  
FORWARD SEARCH command XII-1  
FORWARD WORD command IV-2, IV-14

---

**G**

gathering text XI-8  
GO BACK command II-9

**H**

Headerspacing style parameter XVIII-4, A-27  
HEADING format XVI-2  
headings XVI-1  
hexadecimal numbers A-5, A-9, A-11

**I**

INCLUDE command XVII-4  
INDENT format XIV-25  
Indentation style parameter XIII-9, XVIII-4, A-10, A-28  
INDEX command XVI-9, A-29, B-13  
input and output ports (defining) A-9  
INSERT command V-10  
INSERT FILE command V-12, V-13  
Installation A-1  
Integral Data Systems 460 A-22  
intermediate file A-15  
I/O port characteristics A-8  
ITALICS format XV-3  
ITEMIZE format XIV-18

**J**

Justification style parameter XVIII-4  
Justifying A-28

**K**

Keyword B-4

**L**

Leftmargin style parameter XVIII-4, A-3, A-27  
LEVEL format XIV-12  
Levelhang style parameter XVIII-4  
Levelindent style parameter XVIII-4  
line spacing XVIII-1, A-27  
Linewidth style parameter XVIII-5  
lowercase XII-9

---

**M**

mailing labels A-16  
MAJORHEADING format XVI-2  
margins XIII-9, XVIII-2, A-27  
mark boundaries XI-2, XI-6, XI-7  
MARK SET command V-7, V-8, V-13  
MARK WHOLE PARAGRAPH command V-9, V-13, XI-6, XI-7  
measurements A-15  
memory B-13  
MESSAGE FORMAT command XVII-5  
Meta-Control II-7  
Meta-Repeat VII-5  
micas A-15  
mismatches XII-9  
missing fences XIII-9, B-12  
Mode II-3  
Mode Line II-1, II-3, V-6, XI-8  
Modem A-18  
moving text XI-1, XI-2, XI-4, XI-8  
.mss file extension VII-4  
multiple buffers IX-2  
multiple copies VII-8  
multiple files IX-1  
multipurpose documents XVII-1

**N**

NEC Spinwriter A-21, A-25  
nesting format commands XIII-8  
new buffer IX-7  
new file VI-7, IX-5  
newline III-4, V-11, XI-9, B-3  
NEWPAGE format command XVII-10, B-12  
NEXT LINE command IV-6, IV-14  
Normal Mode VIII-1  
Not Found XII-9  
NOTE format command XVI-11, B-13  
Notes style parameter XVIII-5, A-29  
NUMBERED HEADINGS XVI-1, XVI-4

---

**O**

ONE WINDOW command X-6  
ONE-WORD format XV-5  
OPEN INSERT command V-10  
operating system VI-1  
    see CP/M  
OTHER WINDOW command X-12, X-14, XI  
output device XVIII-1, A-15  
overflow B-13  
Overwrite file VI-7  
Overwrite mode VIII-2

**P**

padding A-5, A-6, A-29  
PAGEFOOTING XVI-12, XVI-13, XVI-14, XVII-10, B-10  
PAGEHEADING XVI-12, XVI-14, XVI-16, XVII-10, B-10  
page space A-29, B-13  
Paperlength style parameter XVIII-5  
Paperwidth style parameter XVIII-5  
paragraph III-4  
PARAGRAPH format XVI-4  
parallel port A-20  
pause VII-8  
personal taste A-10, A-27  
pitch A-21, A-23  
Point II-2  
predefined word variables XVII-14  
PREVIOUS LINE command IV-6, IV-14  
print VII-1  
printer A-13 to A-32  
printer configuration A-13  
Printer definition questionnaire A-23  
proportional-space A-21, A-24 to A-26  
punch A-18

---

**Q**

QUIT command III-3  
QUOTATION format XIV-6, A-28

**R**

READ FILE command VI-5  
Redisplay command IV-13, A-10  
@REF format command XVII-16, B-7, B-8  
REPEAT command VIII-4  
REVERSE SEARCH command XII-3  
Return key III-4  
Rightmargin style parameter XVIII-5, A-27, A-28  
ROMAN format XV-5

**S**

SAVE FILE command VI-5, VI-6  
Save mode VII-1  
save sequential deletions V-6  
Screen display II-1, IV-7  
Scriptpush style parameter XVIII-5  
Scroll IV-7  
SEARCH AND REPLACE command XII-4  
SEARCH AND REPLACE {with Query} command XII-6  
searching XII-1  
SECTION format XVI-4, A-16  
SECTION headings XVI-4  
serial interface A-20  
@SET format command XVII-16, B-8  
SET/REF format XVII-16  
Space bar III-4  
Spacing style parameter XIII-9, XVIII-6, A-27  
split-screen VII-10  
standard file A-15  
STOP printing VII-9  
Spread style parameter XVII-6, A-27  
Storing text VI-1  
STRING format XVII-11, B-8, B-9  
STRING/VALUE format XVII-11  
style parameter commands XIII-9, XVIII-2, B-10  
SUBHEADING format XVI-3  
SUBSCRIPT format XV-4, A-28, A-29  
SUBSECTION format XVI-4  
Successive occurrences XII-2  
SUPERSCRIPT format XV-4, A-28, A-29  
swap file IX-9, A-10, C-1  
SWITCH BUFFERS command IX-6, IX-10, X-15  
Synchronization protocol A-17, A-18  
system messages II-3

---

**T**

tab key III-4  
tab spacing A-10, A-28  
table of contents XVI-1, A-29, B-13  
temporary save buffer XI-1, XI-2, XI-4, XI-8  
temporary workspace IX-1  
terminal A-11, A-12  
Terminal definition questionnaire A-5  
test.mss A-32  
TEXT format XIV-4, A-28  
TITLE format XVII-15, B-9  
tokens B-13, B-14  
Topmargin style parameter XVIII-6, A-27  
translation table A-26  
TRANSPOSE CHARACTERS command VIII-7  
TRANSPOSE WORDS command VIII-7  
TWO WINDOWS command X-1, X-4, X-13, XI-6  
typefaces XV-1  
TYPEWRITER format XV-5

**U**

UNDENT format XIV-24, A-28  
underlining XV-2, A-6  
UNDERLINING formats XV-2, A-6  
UNIVERSAL REPEAT command VIII-4, B-3  
UNNUMBERED format XVI-2  
unnumbered headings XVI-1  
uppercase XII-9  
UPPERCASE WORD command VIII-6

---

**V**

VALUE format XVII-9  
"Vanilla" printer A-19  
Variables XVII-11, B-8, B-9  
VERBATIM format XIV-2  
VERSE format XIV-22  
View mode VIII-2  
VIEW NEXT SCREEN command IV-8, IV-14, A-10  
VIEW NEXT SCREEN (OTHER WINDOW) X-8  
VIEW PREVIOUS SCREEN command IV-8, IV-14, A-10  
VIEW PREVIOUS SCREEN (OTHER WINDOW) X-10  
virtual memory VII-10, A-10, C-1

**W**

Window II-1, IV-7, X-1  
WIPE REGION command V-7, V-8, V-13, XI-2, XI-4  
WRITE FILE command VI-5, VI-7

**X**

Xerox A-21

**Y**

YANKBACK command V-6, V-9, V-13, XI-1, XI-6, XI-9  
Yes/No Responses II-9, A-14, B-12

---

# Notes

## Notes

## Notes





# PERFECT SPELLER<sup>TM</sup>

By Robert B. Wesson, Ph.D.

**USER'S GUIDE**

#### **COPYRIGHT**

Copyright, 1982 by Perfect Software, Inc. All rights reserved worldwide. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system or translated into any human or computer language in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the express written permission of Perfect Software, Inc., 1400 Shattuck Avenue, Berkeley, California 94709.

#### **DISCLAIMER OF WARRANTY**

Perfect Software, Inc. makes no representations or warranties, either express or implied, with respect to this manual and accompanying software and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. This manual and accompanying software are sold "as is" and Perfect Software will in no event be liable for direct, indirect, incidental or consequential damages resulting from any defect, error or failure to perform.

#### **TRADEMARK**

Perfect Writer<sup>TM</sup>, Perfect Speller<sup>TM</sup>, Perfect Mailer<sup>TM</sup>, Perfect Sort<sup>TM</sup>, Perfect Terminal<sup>TM</sup>, Perfect Messenger<sup>TM</sup>, Perfect Calc<sup>TM</sup>, Perfect Ledger<sup>TM</sup>, Perfect Software<sup>TM</sup>, and the Perfect<sup>TM</sup> prefix are trademarks of Perfect Software, Inc.

---

# **PERFECT SPELLER USER'S GUIDE**

- I. Introduction
- II. Using Perfect Speller
- III. About the Dictionary
- IV. Optional Parameters
- V. Customizing Perfect Speller

## **APPENDICES**

- A. Configuring Perfect Speller
- B. Limitations

---

2

---

## Chapter I INTRODUCTION

Perfect Speller is an advanced software spelling checker that will identify and correct typing and spelling errors. Designed to work directly with the Perfect Writer word processor, Perfect Speller is implemented by a single command that causes your document to be checked against a machine readable dictionary equivalent to more than 50,000 words. After finding words that do not match words in the dictionary, Perfect Speller allows you to review the mismatched words, marking those that are actually misspelled, while ignoring those that are not (e.g., proper names). Perfect Speller then takes you to each word you have marked in your document, stopping to allow you either to correct it, add it to the Perfect Speller dictionary, or ignore it.

Perfect Speller can look up about 4,000 words per minute on a typical microcomputer. It typically processes a page of single spaced text in seven seconds and a 20-page document in less than three minutes (compared to more than 10 minutes for the premier IBM Displaywriter).

Perfect Speller is extremely easy to learn. To begin using it to correct your spelling and typing errors, you need only to read the next section, which presents a step-by-step tutorial on using Perfect Speller. The remaining chapters of the manual present the fine points of Perfect Speller's operation. For example, Perfect Speller will allow the mismatched words of a document to be printed either to your terminal or to a file. Lists of new words can be entered from a file directly into Perfect Speller's dictionary. In addition, it is possible to create your own specialized dictionaries, to meet whatever particular needs you may have. (Several specialized dictionaries are currently available for Perfect Speller, including medical, legal, real estate, business, insurance and several foreign language dictionaries.)

---

*I-2 Introduction*

---

---

## Chapter II

### USING PERFECT SPELLER

**(IMPORTANT:** Before beginning this tutorial be sure that you have read Appendix A and have configured Perfect Speller for your particular computer system. For the purposes of this tutorial, we will assume that all needed Perfect Speller and Perfect Writer files are present on your default disk.)

---

#### A TUTORIAL

---

Suppose that you have just finished typing on Perfect Writer the following file, named "letter.mss", and now wish to check for misspelled or mistyped words:

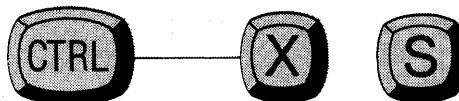
#### Contents of "letter.mss"

Dear Mr. Thompson:

This is a sampel letter that we will check using Perfect Speller. We have intentionally made errors two illustrate how Perfect Speller works. After you create a letter with Perfect Writer you can check its spelling by simpley entering the CHECK SPELLING command. After checking your letter (or othre document) Perfect Speller takes you to each mispelled word and lets yuo correct it. The simplicity of this exercise illustrates that Perfect Speller is truly a 'user friendly' in-context spelling checker.

**STEP 1:**

Enter Spell Mode by typing the CHECK SPELLING command:



(While holding the Control key down, press the "X" key, then release the Control key and press the "S" key.)

**STEP 2:**

Following an initial few seconds that are required to read the large dictionary file from disk to machine memory, Perfect Speller asks for the name of the file to be checked.

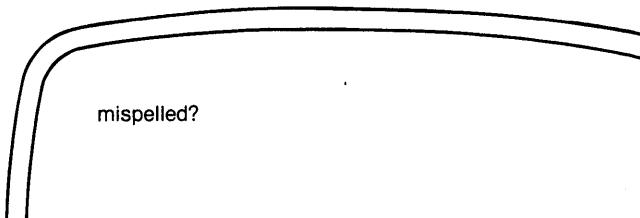
Perfect Speller begins checking "letter.mss" for misspelled words. (This is a process that will take from a few seconds to a couple of minutes depending upon the length of the document.) As it works, Perfect Speller will display the total number of words processed together with the number of words not recognized. For this example:

84 words processed. 6 words not recognized.  
Scan list of unrecognized words now<yes>?

**STEP 3:**

Answering "No" causes Perfect Speller to mark the unrecognized words in your document and to transfer you back to Perfect Writer where you can correct each misspelled word in context.

Answering "Yes" instructs Perfect Speller to begin displaying, alphabetically and one at a time, the words from your document which were not found in its dictionary.<sup>1</sup> For this example, the first misspelled word on Perfect Speller's alphabetized list would be:



---

<sup>1</sup> If you hit the <RETURN> key or carriage return in response to this question, then Perfect Speller will supply the default answer indicated within the brackets.

**STEP 4:**

At this point you have the following command options:

**SCAN COMMANDS**

- a** - ADD word to dictionary. (Select this option for words which are spelled correctly and which you use frequently. Perfect Speller will collect these words and add them to its dictionary at the end of the spelling session.)
- i** - IGNORE word. (Use this response for those words which are spelled correctly, but which you do not want to add to the dictionary—e.g. proper names.)
- c** - Mark word to CHANGE. (Select this option for words which are actually misspelled or for words which you would like to examine in context.)
- r** - Enter the word's ROOT into the dictionary. (If the word contains a prefix or suffix, use this response to enter its root into the dictionary. Perfect Speller will then be able to recognize not only that particular word but many of its derivations as well. Perfect Speller will ask you to enter the word root, followed by a carriage return.)
- e** - Mark remaining words and EDIT text. (Use this option to discontinue scanning the words. Perfect Speller transfers you directly to your document to correct the words in context.)
- ?** - Help! (Should you be in doubt what option to choose, this option will cause Perfect Speller to display all of its command options.)

**STEP 5:**

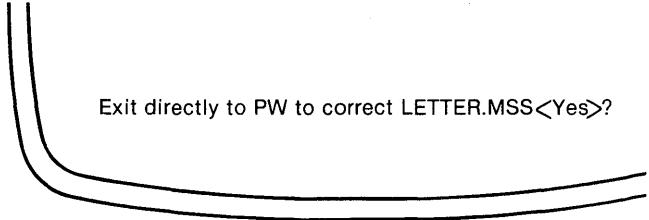
Since *mispelled* is actually *MISSpelled* you would select option "c - Mark word to CHANGE."

Perfect Speller automatically marks the word in your text and displays the next misspelled word from its list. The entire list of misspelled words for the document "letter.mss" would be:

	ACTION	REASON
mispelled?	c(hange)	{misspelled}
othre?	c	{mistyped}
sampel?	c	{mistyped}
simpley?	c	{misspelled}
Thompson?	i(gnore)	{proper name}
yuo?	c	{mistyped}

After you have finished reviewing the list of misspelled words Perfect Speller asks:

Exit directly to PW to correct LETTER.MSS<Yes>?



**STEP 6:**

If you answer "No," Perfect Speller will return you to the Perfect Writer Main Selection Menu.

Answering "Yes" transfers you to your document in Perfect Writer, the cursor positioned at the first word that you have marked for change during "scanning."<sup>2</sup> Notice that all of the words that are misspelled have been marked with the sign "~~S". Only the word before where the cursor is does not display this mark. The cursor will erase the mark as it moves from word to word.

Dear Mr. Thompson:

This is a sampel letter that we will check using Perfect Speller. We have intentionally made errors two illustrate how Perfect Speller works. After you create a letter with Perfect Writer you can check its spelling by ~~ Ssimply entering the CHECK SPELLING command. After checking your letter (or ~~ Sothe document) Perfect Speller takes you to each ~~ Smispelled word and lets ~~ Syuo correct it. The simplicity of this exercise illustrates that Perfect Speller is truly a 'user friendly' in-context spelling checker.

---

<sup>2</sup> As indicated earlier, hitting <RETURN> will cause Perfect Speller to provide the default value indicated within the brackets.

**STEP 7:**

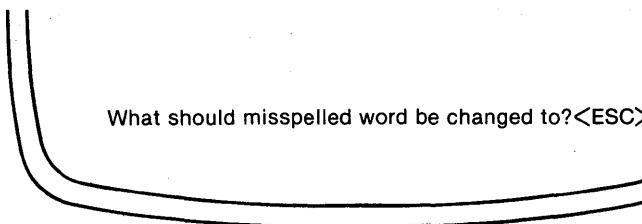
At this point your options are essentially the same as before, except that now you are actually going to be correcting your document. The "in-context" correction commands are the same as before:

**IN-CONTEXT CORRECTION COMMANDS**

**A** - Add, **C** - Change, **E** - Edit, **I** - Ignore, **R** - Root, **?** - Help!

**ADD** Selecting "add" causes the word to be added to Perfect Speller's dictionary.

**CHANGE** IF you select "change," Perfect Speller responds:



Enter the correct spelling of the word followed by Escape. Perfect Speller replaces the misspelled word with the corrected word you have just typed afterwards positioning the cursor at the next misspelled word.

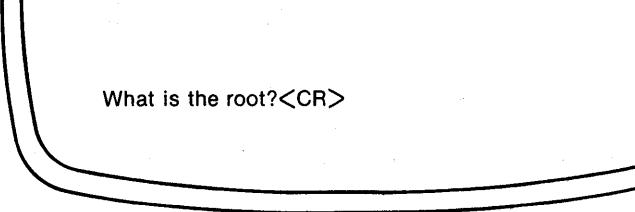
**EDIT** If you select "edit," Perfect Speller switches from Spell Mode to Normal Mode in Perfect Writer, thus allowing you to edit the file, changing the word and sentences as you wish. The remaining misspelled words remain marked to await your return to Spell Mode, which you can do by using the ADD MODE command (Control ----- X M and entering "spell").

**IGNORE** Selecting "Ignore" causes Perfect Speller to skip to the next marked word, leaving the present marked word unchanged, but no longer marked for correction.

---

**ROOT**

"Root" indicates that you wish to enter the root of this word into Perfect Speller's dictionary. Perfect Speller responds to this command with:

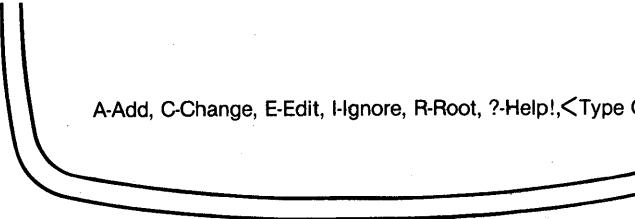


What is the root?<CR>

Type the root of the word followed by a carriage return. (For a discussion of word 'roots,' see Chapter III, page 2.)

**HELP!**

Select "help" and Perfect Writer displays the options currently available to you, namely:

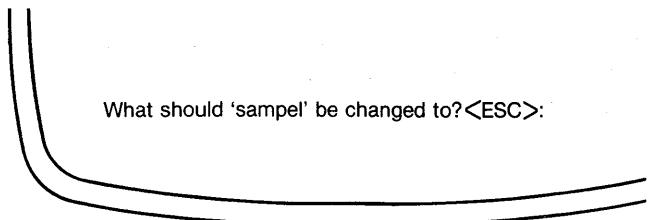


A-Add, C-Change, E-Edit, I-Ignore, R-Root, ?-Help!,<Type CR Now>

**STEP 8:**

The cursor stands before *Sampel*, the first misspelled word that we marked in our document. To correct this word we select option "C-Change."

Perfect Writer responds in the Mode Line with:



What should 'sampel' be changed to?<ESC>

**STEP 9:**

Enter the correct spelling, *sample*, followed by the Escape key.

Perfect Speller automatically changes *sampel* to *sample* in the text, afterwards moving the cursor to the next misspelled word, *simpley*, which would be corrected in the same fashion.

Notice that Perfect Speller did not identify the improper use of *two*, in the sentence, "We have intentionally made errors *two* illustrate...etc."

**STEP 10:**

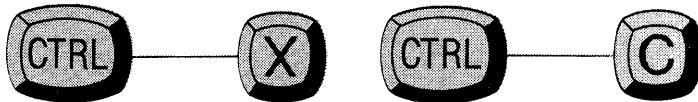
After you have reviewed and corrected all misspelled words you can either replace the old document with the corrected version using the SAVE FILE command:



or you can save the corrected version under a new filename using the WRITE FILE command:



After saving, your document is ready for output to the printer, which can be selected by returning to the main menu of Perfect Writer, using the QUIT command:



---

END OF TUTORIAL

---

*II-10 Using Perfect Speller*

---

## Chapter III

### **ABOUT THE DICTIONARY**

If any words have been added to the dictionary, Perfect Speller writes the new, expanded dictionary to disk. In this process the old dictionary is normally deleted after the new one has been written. However, sometimes it happens that there is not enough room on disk to contain both the old and new dictionaries at once. In such a case Perfect Speller will offer you the opportunity of deleting the old dictionary first. Perfect Speller warns you that this is risky, since there is the danger that if the write operation should fail to complete, the dictionary will be lost. Naturally, you keep a backup copy of your important files (like the dictionary) on a separate disk, and so the danger only amounts to the inconvenience of having to recover your dictionary from your backup. In such an event, normally you will want Perfect Speller to go ahead and delete the old dictionary before writing out the new one and can answer yes to the "delete old dictionary..." offer.

When adding words to its dictionary, Perfect Speller uses a method which keeps the file the same size, but which increases very slightly the chance that Perfect Speller will miss an incorrect word. If, for example, you add a thousand words to the dictionary, you increase the chance of letting a misspelled error go undetected by roughly 1/20th of one percent (a probability equal to .0005). It should be understood that the chance of this happening increases the larger the dictionary becomes. For example, the following table shows the estimated frequency of undetected errors with the dictionary provided, together with the effect of adding 1,000 words to the dictionary.

---

<b>Number Missed Errors in 1,000 Errors</b>	<b>Number if 1,000 Words Added to Dictionary</b>
DICTIONARY.SPL	2.2

---

Given these figures, you should use some restraint in adding words to the dictionary. Our experience has been that most users typically add a few hundred words or so at most, so this should be a minor constraint. Remember—if you need a dictionary for a specialized field such as law or medicine, call us. We've probably already constructed it. If you really do need to add large lists of words to the dictionary, please see Chapter V.

### **What is a Root?**

A root is the basic element of a word from which a great many other words can be derived by the addition of prefixes and suffixes. For example, consider the word *restored*. *Restored* consists of the prefix *re-*, the root *store*, and the suffix *-ed*.

Perfect Speller already knows most possible prefixes and suffixes for English words. Therefore when a word like *restored* needs to be added to the dictionary we add only the root, i.e. *store*. This will enable Perfect Speller to recognize not only *restored*, but also words like *restore*, *storing*, *storage*, and *restoration*.

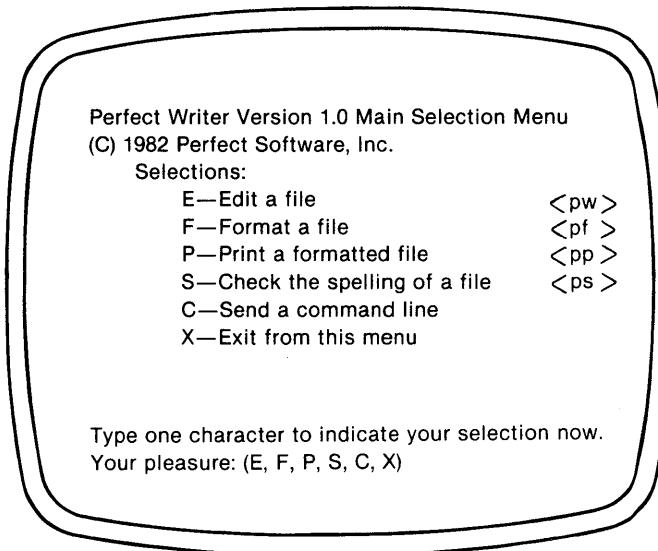
Some words, of course, do not have suffixes and prefixes that can be stripped away to reveal a root. In these cases, the word is unique and must be entered entirely.

## Chapter IV **OPTIONAL PARAMETERS**

### **Access Through the Menu**

The tutorial in Chapter II illustrates the basic steps in using Perfect Speller to correct spelling errors in a document. The procedure is started simply by giving the CHECK SPELLING command, Control ---- X S, while editing in the Perfect Writer editing mode.

Perfect Speller can also be invoked through Perfect Writer's Main Selection Menu, a procedure which offers slightly different options, and which we will discuss now. As you recall, Perfect Writer's Main Selection Menu looks like this:

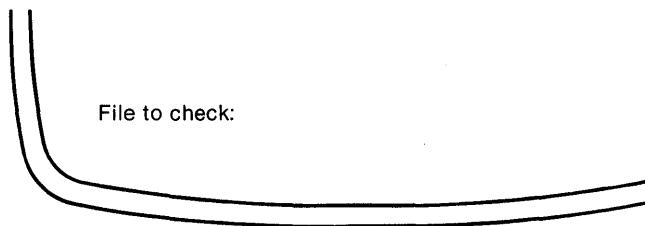


## *IV-2 Optional Parameters*

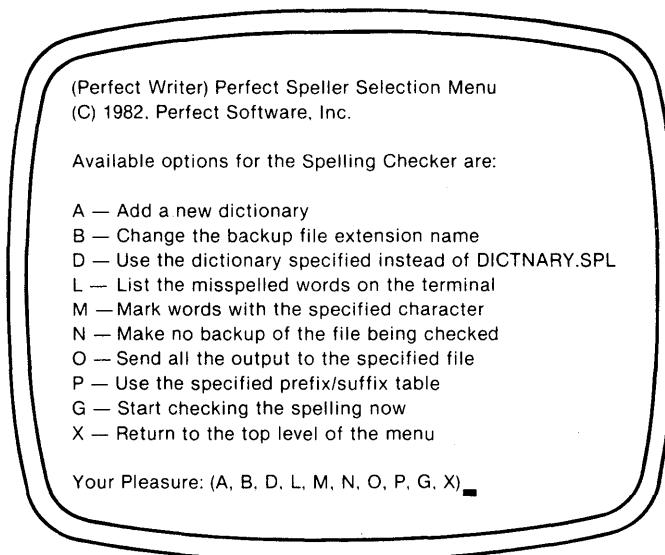
---

Perfect Speller is represented in the menu as option "S—Check the spelling of a file."

Selecting this option causes Perfect Writer to respond with the message:



After you enter the filename, you are transferred to Perfect Speller's Selection Menu, which presents the various options available to you with Perfect Speller. Note that in the tutorial these options were completely bypassed by the CHECK SPELLING command (which is equivalent to option "G—Start checking the spelling now.")



## **A—ADD A NEW DICTIONARY**

Select this option when you wish to create a new dictionary, either from scratch or by significantly altering the existing dictionary, "DICTNARY.SPL" (see Chapter V).

## **B—CHANGE BACKUP EXTENSION NAME**

It is possible to change the ".BAK" filename extension of the backup file that Perfect Speller creates before marking the spelling errors in your document. This can be done when necessary to avoid confusing the backup with other existing files that possess the same file extension.

## **D—SELECT DICTIONARY**

If you have created other dictionaries besides DICTNARY.SPL (or have purchased any of the specialized dictionaries), these can be selected using this option. If you use this option from the\*DOS command line, you should follow it with the filename of the dictionary you wish to use. If you select this option from the MENU, then you will be prompted for the filename of the dictionary.

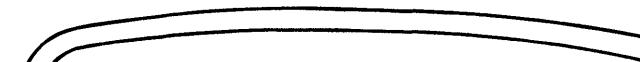
---

\*DOS stands for Disk Operating System (i.e., CP/M, MSDOS, CP/M86, etc)

## L—LIST WORDS TO TERMINAL

This option causes Perfect Speller to quickly scan a document for unknown words, afterwards listing them to your terminal in alphabetical order. Your document is not marked, and no backup file is created. You are not questioned or allowed to change or alter mistakes. This option is often used in conjunction with option 'O' (see below).

This option can be selected to perform a very quick scan over short documents, or for doing quick dictionary lookups in general. For example, from DOS, you might have a short note which you wish to quickly check for misspellings. You anticipate a few unrecognized words, but most of them are likely to be proper names which can be ignored anyway. Your DOS terminal interaction could look something like this:

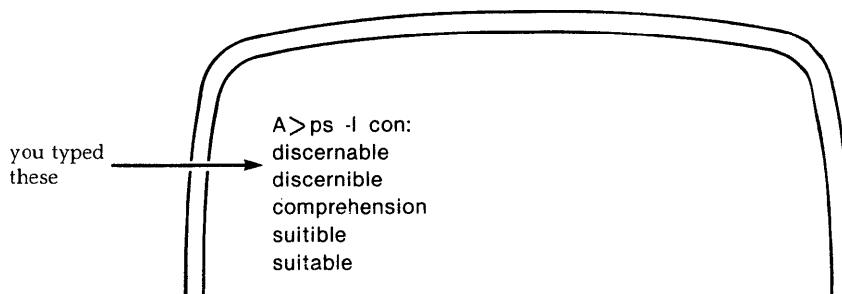


```
A> ps -l note.mss
Allentown
McCaulley
processer
Westwood
```

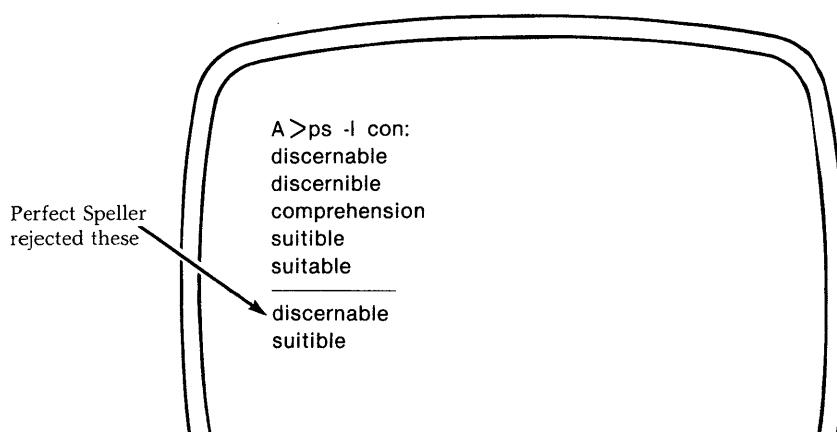
As you can see, if the 'L' option is selected, Perfect Speller simply lists the unrecognized words out to your terminal, but does no further processing. You will have to activate Perfect Writer to correct this single misspelled word found above (*processer*), but remembering one or a few such words is relatively easy.

---

Another use of the 'L' option is to use Perfect Speller as a quick on-line dictionary lookup program. You can simply select the console as the input file to Perfect Speller, and type in your word list of questionable words. Perfect Speller will list those words which are not recognized, thusly:



terminate  
your input  
with a  
Control ----- Z<sup>3</sup>



Whenever the 'L' option is selected, Perfect Speller suppresses its header line and all questions. This option, in conjunction with the redirected I/O (input/output) discussed below (option 'O'), is intended for the programmer or others who wish to use Perfect Speller as a software tool as well as part of a word processing system.

---

## **M—CHANGE MARKER CHARACTER**

When during scanning, you mark misspelled words for later correction, Perfect Speller flags each word with a 'tilde-uparrow-S' ('`^S). This is a specially coded character that cannot be accidentally entered from the keyboard (thus disallowing inadvertent marking of your document). It is possible, using this option, to change this marker to any character you wish. Simply select "M" and supply the character when Perfect Speller asks for it.

## **N—NO BACKUP**

As was mentioned previously, Perfect Speller, before marking your document, always makes a backup of the document, tagged with the file extension ".BAK." Normally this backup is saved until the entire spelling session has been completed, to guard against accidental losses of your original material. This option causes the backup file to be erased immediately after the marked file has been created, thus saving you potentially vital disk space for other computer functions.

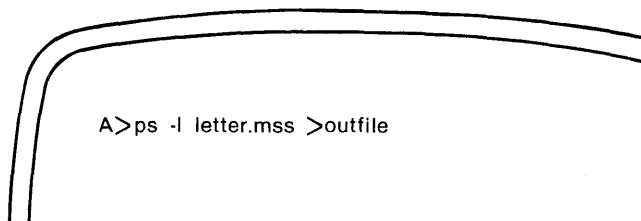
Whenever Perfect Speller is called directly from within the Perfect Writer editor (and not from the menu or DOS), this option is automatically selected for you, since it is assumed that you will return immediately to the editor to correct your document and will have no need for a backup version with (possibly) misspelled words.

## O—PRINT OUTPUT TO FILE

This option instructs Perfect Speller to send the entire output of a spelling session—everything which would have normally been typed to your terminal—to a disk file.

This feature should only be used with the 'L' option, which, as you may recall, sends the entire list of unrecognized words to your terminal. If you combine the two options, Perfect Speller will place this list of misspelled words into a text file for later reference. If you use the 'O' option without the 'L' option, the questions Perfect Speller normally asks during its word disposition phase would be sent to the disk file, leaving you with a running program requesting terminal input from you but without your knowing what it wants.

This feature of Perfect Speller is best characterized as an "advanced" function which should be used only when you have sufficient experience with Perfect Speller in particular and your operating system in general.<sup>4</sup>



---

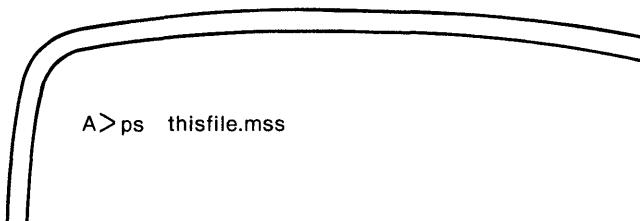
Redirected input is also allowed, but its utility is questionable. Its syntax is identical to that of the redirected output, except that the input filename is prefixed with a '<' character.

## ACCESSING PERFECT SPELLER FROM DOS

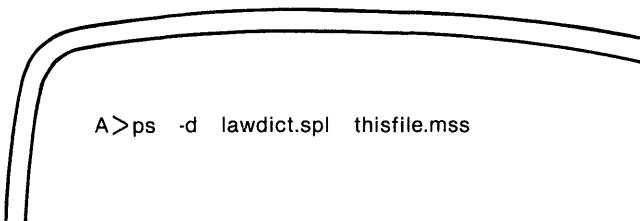
Options that you select from the Perfect Speller menu are automatically displayed in the menu Echo Line.

As in Perfect Writer, these symbols represent the command form that can be given directly from the DOS operating system, thus circumventing the menu altogether. The following examples show various commands to Perfect Speller that might be given:

1. Check the spelling of "thisfile.mss"



2. Check the spelling of "thisfile.mss" using the dictionary "LAWDICT.SPL":



3. Check the spelling of "thisfile.mss" using LAWDICT.SPL (option 'D'), listing the misspelled words to the screen (option 'L').

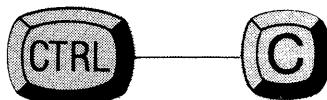
A>ps -l -d lawdict.spl thisfile.mss

4. Write the misspelled words in "thisfile.mss" to a file called "errors.mss":

A>ps -l thisfile.mss >errors

## **Cancel**

You may cancel Perfect Speller's processing at any time and return to the system by typing:





## Chapter V

# CUSTOMIZING PERFECT SPELLER

### **Creating new dictionaries**

To meet their particular needs, most users can simply extend the supplied dictionary file DICTNARY.SPL through routine operation of Perfect Speller as previously discussed. Nevertheless, if you want to create your own dictionary from scratch, you can do so with Perfect Speller. This chapter discusses the procedure.

When creating a new dictionary, Perfect Speller utilizes an intermediate file called "ADDDICT.SPL," the purpose of which is to collect new words prior to their inclusion in the new dictionary. ADDDICT.SPL is automatically created and maintained by the Perfect Writer text editing system. Whenever you want to create new dictionaries, you must first create an ADDDICT.SPL file containing your own words. The following two tutorials demonstrate how to do this.

Another main factor to be considered in creating a new dictionary is size. If a dictionary is too large, it will either not fit into your machine memory at all, or it will not leave enough room for Perfect Speller to keep its alphabetized list of misspelled words. On the other hand, a dictionary that is too small will yield poor accuracy. Perfect Speller was designed to work on a 56K or larger computer. The optimal size of DICTNARY.SPL has been found to be 29,952 bytes.

Although you need not, it is recommended that you follow this standard size in creating new dictionaries. Whatever dictionary size you choose (in bytes), it must be a multiple of 256 and cannot be larger than 32,767.

Following are two procedures for creating a new dictionary. The first, and recommended, procedure operates by extending the existing DICTNARY.SPL; the second by creating a completely new dictionary from scratch.

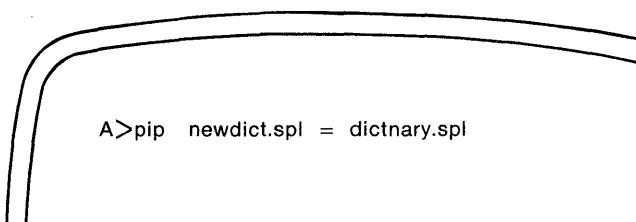
---

## A TUTORIAL

### Creating a New Dictionary from "DICTNARY.SPL"

#### STEP 1:

Using DOS copy the main dictionary DICTNARY.SPL into a new file, say "NEWDICT.SPL."



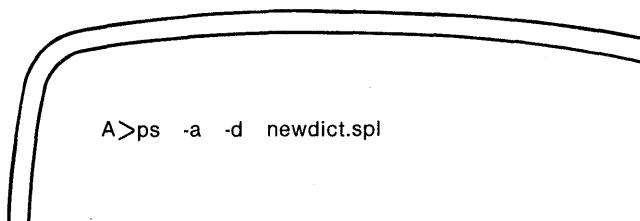
("NEWDICT.SPL" will be the new dictionary.)

#### STEP 2:

Using Perfect Writer, enter into the file ADDDICT.SPL all of the words that will be added to NEWDICT.SPL. Use either upper or lower case letters, entering the words one to a line, or together on the same line, separating them by a space or any non-alphanumeric character (except a hyphen, which is considered to be part of a word).

#### STEP 3:

From DOS invoke options 'A' and 'D' of Perfect Speller:



Your new words in ADDDICT.SPL will automatically be added to the new dictionary NEWDICT.SPL. Note that Perfect Speller will delete the file ADDDICT.SPL after adding its contents to the dictionary, so if you wish to save your wordlist, you should have saved another copy of it.

Two dictionaries will exist on the original system disk: NEWDICT.SPL and DICTNARY.SPL. In checking a document Perfect Speller will always use DICTNARY.SPL, unless you select NEWDICT.SPL using option 'D.'

**\*NOTE:** In modifying DICTNARY.SPL in this way, not all of the new words will likely be added, since many of them probably exist in DICTNARY.SPL to begin with. Perfect Speller adds only those words not already present and ignores the rest.

**\*\*ALSO,** any time DICTNARY.SPL is significantly enlarged, the chance for error in detecting misspelled words is increased (see Chapter III). Restraint should therefore be used in adding words to the main dictionary. In general, the addition of as many as 5,000 words to DICTNARY.SPL will not increase Perfect Speller's error rate beyond acceptable bounds.

---

**End of Tutorial**

---

---

## A TUTORIAL

### Creating a New Dictionary

**STEP 1:**

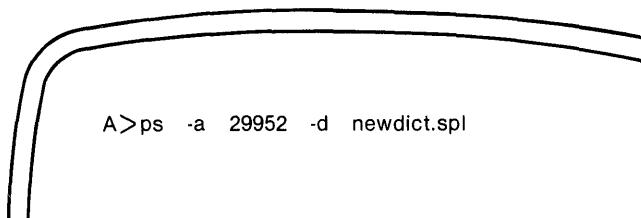
Decide the size of the new dictionary in bytes. It must be a multiple of 256, and must not exceed 32,767.

**STEP 2:**

Create the file ADDDICT.SPL containing ALL of the words that will make up the new dictionary. These words can be entered either one to a line, or one after the other on the same line, separated by a space or any non-alphanumeric character (except a hyphen). Both lower and upper case letters are acceptable.

**STEP 3:**

Invoke the 'A' option of the Perfect Speller menu, following it with the length the dictionary will be. The menu Echo Line will reflect your selection along with the length you specify. For example:



(In this example, you are selecting a different dictionary than the default DICTIONARY.SPL.)

**STEP 4:**

Perfect Speller will automatically create a new dictionary file NEWDICT.SPL and encode the words from ADDDICT.SPL into it. The original dictionary file DICTIONARY.SPL remains unchanged, and ADDDICT.SPL will be deleted as usual.

---

End of Tutorial

---

## Modifying the Prefix/Suffix Table

The file holding the Prefix and Suffix Table used by Perfect Speller is called "AFFIXTAB.SPL." You may add or delete prefixes or suffixes from AFFIXTAB.SPL to customize Perfect Speller for your particular spelling needs. For example, you may prefer the British spellings of words instead of the American (e.g., *civilised* instead of *civilized*). To inspect and modify AFFIXTAB.SPL, call the file into Perfect Writer's editing mode. As you will see, AFFIXTAB.SPL is simply a list of prefixes and suffixes one per line. The prefixes are listed first in the file followed by the suffixes. It is possible to add about 40 suffixes and 20 prefixes to this list if you wish, more if you delete some. Use lower case letters only, except for "C" as mentioned below. Order is not important, except that longer prefixes must follow shorter prefixes which contain them; e.g., *under* must follow *un*.

Note that suffixes are represented as "- characters deleted + characters added," so that *mercy* becomes *merciless* via the application of the suffix "-y+less." An uppercase "C" indicates a final consonant, so that the rule "double the final consonant and add 'er'" is encoded as +C+er. Perfect Speller performs limited error checking when reading AFFIXTAB.SPL, so you should be careful if you modify it.

To cause Perfect Speller to accept British spellings, you may add suffixes such as +*re*, -*or*+*our*, and +*ise*.

**V-6 Customizing Perfect Speller**

---

## Appendix A **CONFIGURING PERFECT SPELLER**

The Perfect Speller system consists of the following files:

---

PS.COM	The Perfect Speller Program
DICTNARY.SPL	The Dictionary
AFFIXTAB.SPL	The Prefix/Suffix Table

---

There is no particular "installation" procedure required for you to begin using Perfect Speller. Of course, you should back up the distribution disk as you have undoubtedly already learned to do with your other important programs. After copying the three Perfect Speller files onto another working disk, you may use Perfect Speller immediately with no further ado.

If you have sufficient disk space available (at least 150k per drive), we recommend that you keep the Perfect Writer editing system and Perfect Speller together on the same disk in drive A. You then have all of disk drive B for your document. When you become adept at using DOS and all the elements of the Perfect Writer system, you will probably want to delete the file MENU.COM from your working disk and free up additional disk space.

If you have a minimal system with limited disk storage (less than about 150k per disk drive), you may have to do some juggling with the files in the Perfect Writer and Perfect Speller system to get everything on-line at once. You may have to split up the auxiliary files (DICTNARY.SPL, AFFIXTAB.SPL, and PW.SWP) from their respective main programs (PS.COM and PW.COM), for example. You may wish to have an editing disk, a spelling disk, a printing disk, and a document disk. Or you may wish to put the auxiliary files on drive A, the main programs PS.COM and PW.COM on drive B, and your document (assuming it is relatively short) wherever it fits. In performing this juggling act with your small disk drives, just remember the following rule:

Each main program looks for its auxiliary files first on the default disk and then on drive A.

If you must separate the various files keep in mind the search strategy above and select your default drive accordingly.

## Appendix B **LIMITATIONS**

Obviously Perfect Speller is not a complete text proofreading system, and therefore is not without limitations.

It cannot detect every type of error you make. For example, it cannot check usage. It cannot tell you whether your use of the word *affected* should really be *effected*, or whether you typed *rather then* for *rather than*.

While Perfect Speller knows common prefixes and suffixes, it has little knowledge about where and how to use them. If you use *buyed* instead of *bought*, Perfect Speller will accept it as past tense of the verb *to buy*. If you say *bater* instead of *batter*, Perfect Speller will not complain, since its suffix rules allow for the simple addition of *er* as well as the final consonant doubling case. Thus, while Perfect Speller will spot typographical errors easily, it is not a substitute for knowledge of proper English.

Perfect Speller does not know all the words in the language. You may need to add uncommon words that you use frequently, especially words in technical or specialized fields.

Further, Perfect Speller will not catch typing errors involving non-alphabetic characters, such as digits or punctuation. The only exception to this is the hyphen which is considered an integral part of the spelling of many English words.

Finally, because of the method used to encode its dictionary, Perfect Speller will very occasionally let a misspelled word slip through. This error rate is quite low, however, barely exceeding two misspelled words for every 1,000 misspelled words detected (see Chapter III for a more complete discussion of this).

---

## SYSTEM CONSTRAINTS

Because Perfect Speller collects all unrecognized words in memory, in alphabetical order, it can only handle a few hundred unrecognized words at once. In processing very large documents, Perfect Speller may terminate the dictionary lookup phase early, announcing that its word list is full. Presumably, during the following scan and correction phases you will add some words to the dictionary and correct others, so that a second pass through Perfect Speller will find the remaining misspelled words.

This "graceful exit" philosophy has been applied throughout Perfect Speller. For example, should Perfect Speller encounter a problem writing the newly marked document prior to in-context correction, it will nevertheless list the words which you had marked as misspelled on the terminal screen. Thus, although your text will not be marked (perhaps because you had it on a write-protected disk), your work with Perfect Speller is not completely lost.

All errors which might occur fall into two classes and are, for the most part, self-explanatory:

1. FATAL errors terminate Perfect Speller completely and send you back to the menu or the DOS operating system. Problems reading the dictionary (because of a faulty disk) will cause a FATAL error, for example.
2. WARNING errors indicate that Perfect Speller has detected something wrong but will continue operating, although perhaps in a reduced capacity. Unrecognized switches on a DOS command line will generate a WARNING.

T.M.

Perfect Software, Inc. 1400 Shattuck Avenue Berkeley, California 94709

