GUIDE TO EFFECTIVE SOFTWARE TECHNICAL WRITING

CHRISTINE BROWNING

GUIDE TO EFFECTIVE SOFTWARE TECHNICAL WRITING

CHRISTINE BROWNING

PRENTICE-HALL, INC., *Englewood Cliffs, New Jersey 07632*

Library of Congress Cataloging in Publication Data

Browning, Christine.
Guide to effective software technical writing.

Includes index.
1. Electronic data processing documentation.
2. Technical writing. I. Title.
QA76.9.D6B76 1984 808'.066001 84-6788
ISBN 0-13-369463-1
ISBN 0-13-369455-0 (pbk.)

Editorial/production supervision and interior design: **Karen Skrable**
Cover design: **Jeannette Jacobs**
Manufacturing buyer: **Gordon Osbourne**

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-369463-1
ISBN 0-13-369455-0 {PBK} 01

PRENTICE-HALL INTERNATIONAL, INC., *London*
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, *Sydney*
EDITORA PRENTICE-HALL DO BRASIL, LTDA., *Rio de Janeiro*
PRENTICE-HALL CANADA INC., *Toronto*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
PRENTICE-HALL OF SOUTHEAST ASIA PTE. LTD., *Singapore*
WHITEHALL BOOKS LIMITED, *Wellington, New Zealand*

PREFACE

Computers are one of our most dynamic and fastest-growing modern industries. But how can you become a part of and even play a key role in this environment without being a computer programmer, mathematician, or engineer?

One part of the computer industry that is open to people with a variety of backgrounds is documentation.

> Manufacturers cannot sell computers, software (programs), or accessories unless they provide manuals that tell the customers how to use these products.

The manuals are *not* written by the computer designers and programmers. Even if these technical experts had the time, they would not necessarily have the literary expertise to do the job.

If you

have good communication skills,
believe you can develop a sound writing craft,
are interested in computers,

are willing to learn about the software, and

enjoy helping people to learn,

consider software technical writing, one of the most profitable and rewarding careers in the computer industry.

This book describes how to write good software manuals. It does not assume a strong background in computer technology, but it does assume a strong desire to learn. Writing any type of software manual requires an understanding of the subject and an ability to communicate effectively with the software expert who supplies the information.

- If you are a software technical writer already, this book can help you improve your skills.
- If you are a technical writer who is planning to move into the field of software technical writing, this book can teach you mechanics that minimize the writing effort.
- If you are interested in exploring the possibility of becoming a software technical writer, this book will show you the basic principles of manual writing. Many of these principles also apply to writing computer-related material, such as brochures, promotional literature, and magazine articles.

Chapter 1 introduces software manual writing. It describes why manuals are important, explains their functions, and gives the criteria for writing them.

Chapter 2 describes how to organize the text, establish the readership, and plan the writing task.

Chapter 3 gives a step-by-step approach to writing a software reference manual.

Chapter 4 gives a step-by-step approach to writing a software user manual.

Chapter 5 discusses techniques for structuring a manual to increase its usability.

Chapter 6 presents a style guide. It explains how to ensure readability and avoid common mistakes that detract from the usability of a manual.

Chapter 7 describes the front matter. This includes the revision record, preface, table of contents, and syntax conventions.

Chapter 8 describes the back matter. This includes the appendixes, glossary, and index.

Chapter 9 is an example to test your comprehension.

The glossary provides definitions of computer industry terminology.

Skilled technical writers can play a key role as intermediaries between users and technical experts. The computer will never fulfill its potential unless people can learn to appreciate it and use it effectively. The challenge of providing the required teaching material is open to anyone who finds it exciting and worthwhile.

ACKNOWLEDGMENT

Computer systems, with their text-editing facilities, disk and tape units, and high-speed printers, simplify the writer's task. For making their special equipment available to me, I am deeply grateful to Tandem Computers Incorporated, Cupertino, CA.

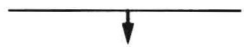
Christine Browning

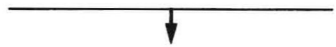
CONTENTS

1/WHAT IS A SOFTWARE MANUAL?

Organizations associated with the computer industry market a variety of software products. Each of these products requires at least one manual that describes its design and operation.

Airlines, for example, use software products that maintain their flight reservation data. Suppose you are taking a trip.

The reservation agent books your flight
by using a computer terminal

to access computer software

designed and coded by analysts and programmers

who use software manuals

to do their jobs.

What happens if the manuals are not complete and accurate?

The analysts and programmers make mistakes,
the computer software does not work the way it is supposed to,
the computer terminal delivers incorrect information, and
you are booked on the wrong flight!

If this gives you some idea of how important software manuals are, consider the manuals that describe software for air traffic control. It is better for you to miss a flight than for the pilot to miss a runway.

WHO WRITES THE MANUALS?

Software manuals are written by specialized writers called software technical writers. These individuals have writing skills, a general knowledge of computer hardware, and a good knowledge of computer software. They take complicated subjects, organize them into logical pieces of information, and produce software manuals that serve as reference or learning tools.

The writers are often writing about new software or new software features with which they are not necessarily familiar. They are often writing the manuals at the same time the software is being developed. How can they do this? By seeking out information from the software designers and developers and by drawing on past experience—similar products, similar concepts.

Software technical writers fall into one of three categories:

1. The junior writer adds update material to existing software manuals, doing relatively little new or original writing.
2. The intermediate writer adds update material to existing software manuals and incorporates a certain amount of new and original writing.
3. The senior writer designs and writes new software manuals.

Each type of writer plays a critical role in the computer industry. Each has a responsibility to write clearly, concisely, accurately, and—always important—quickly.

The customer is eager to have the software, and the developer is eager to deliver it. But the software will not go anywhere without a manual that explains it.

WHAT DOES A SOFTWARE MANUAL LOOK LIKE?

A software manual has its own special format. Formats vary from company to company, but they generally look something like the one shown in Figure 1-1.

- The manual is divided into sections.
- The section carries a running head that identifies the subject matter.
- The section is numbered.
- The section title identifies a major area.
- The first-order paragraph head identifies a major topic within the area.
- The second-order paragraph head identifies a topic subordinate to the first-order paragraph head, and so forth.
- The manual is paginated by section. Pages can be added to one section without forcing other sections to be repaginated.

Breaking down information into these logical units serves several important purposes:

1. Information parallels the structure of the software product and helps readers to learn. A paragraph head *Control Statement Parameters* would probably be followed by subordinate paragraph heads for each of the parameters. Readers first learn that control statements have parameters, and then they find out exactly what those parameters are.
2. The material can be easily maintained. A software manual is usually updated rather than rewritten each time the product changes. The paragraph heads provide logical divisions for adding new or changing existing information. If the product is changed to include three new control statement parameters, the next writer already has a convenient place to put them.

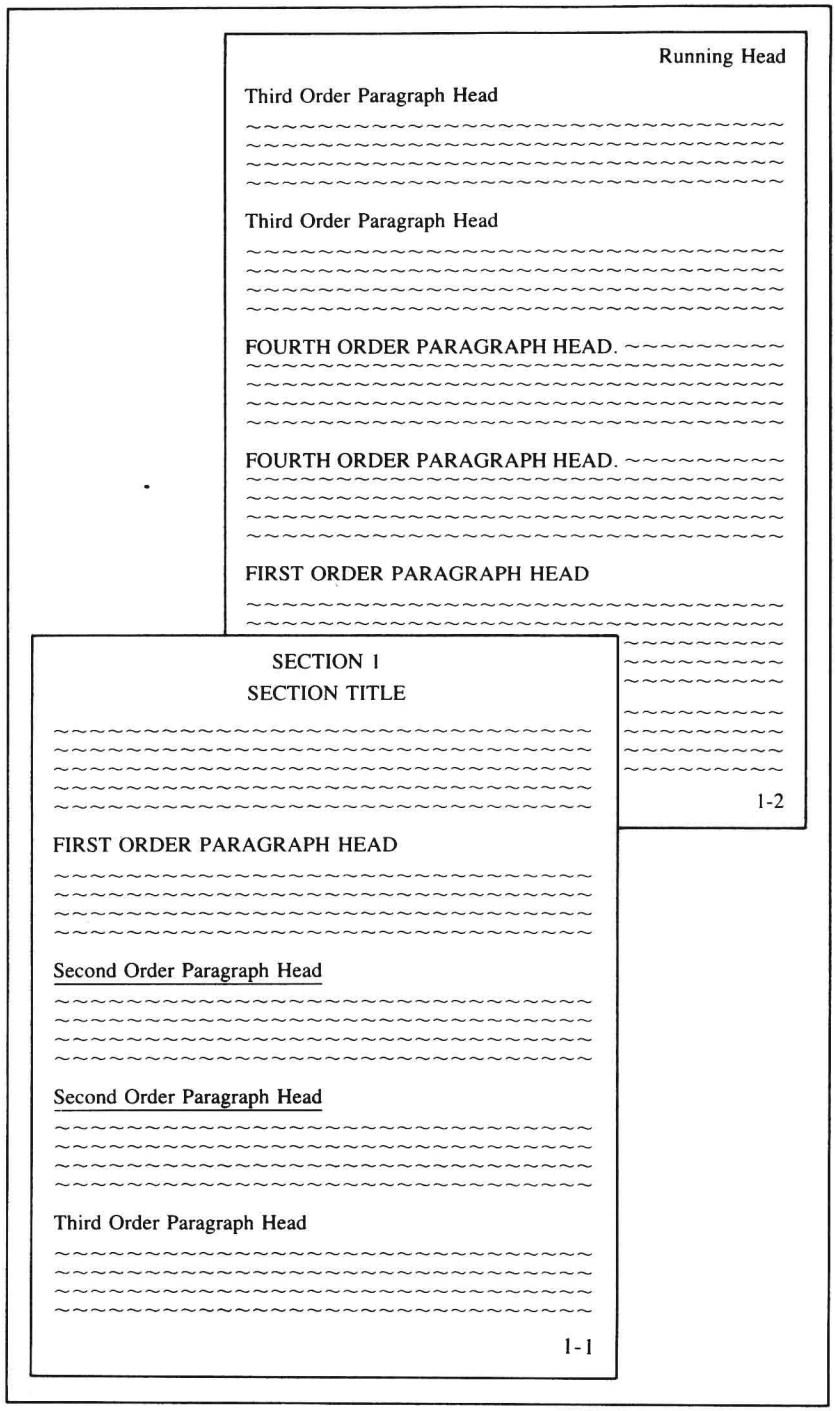


FIGURE 1-1 Software Manual Format

3. Readers are alerted to subject matter and can skip over the familiar or concentrate on the unfamiliar. The paragraph head *Control Statement Parameters* and its subordinate information can be safely ignored by the reader who is not planning to use control statements.

The Appearance of the Manual Is Important

A good novel that consists of page after page of unbroken text holds the reader's interest. The novel has an exciting plot.

A good software manual that consists of page after page of unbroken text does not hold the reader's interest. The manual has no plot.

Good software manuals have text that is broken up in several ways:

- indented text
- indented text preceded by bullets, as shown here
- short paragraphs
- programming examples, often printed in a different type font
- tables
- numbered steps
- illustrations

Figure 1-2 shows the difference between unbroken and broken text. If you had your choice of learning information from one of these sample pages, which page would you select?

The Content of the Manual Is More Important

A manual must do more than look good. To be effective and reflect professionalism, a manual should meet the following criteria:

1. *Material is well organized.* The manual is organized to serve the reader, not to satisfy the writer. A manual is always rated by its usability rather than by its literary style.

The SCANNER and PROC routines perform polling and character processing operations. SCANNER polls all lines in the network and upon detection of an input character, performs a series of tests to determine the legality of the input character. In addition, SCANNER calls a special-purpose routine to set pointers to four core-resident buffers: Term Buffer, Verify Buffer; Format Buffer, and Data Buffer. Term Buffer reflects terminal status; Verify Buffer maintains current record processing parameters; Format Buffer holds the memory address where the format is stored; and Data Buffer holds the input characters until they comprise a full record and require transmission. The pointers are established when the first input character is moved into the Data Buffer.

After the last input character has been verified, SCANNER calls the PROC routine. PROC checks the input characters against the stored format, interprets the operation to be performed, and then transfers control to the appropriate processing routine.

When the polling cycle is complete control passes to the EXECUTIVE program. If an input/output operation is required, the EXECUTIVE calls the appropriate driver, sets the I/O request word to zero, and returns control to SCANNER to continue the polling operations.

Term Buffer

The Term Buffer, which is allocated in ring 5, is a 2-word area that reflects the status of the terminal. Bit positions in word 1 are set to provide specific information regarding terminal activity. Bit 0 is set for input/output, bit 1 for read, bit 2 for write, bit 3 for error recovery, and bit 5 for parity checking. The remaining positions are reserved for future expansion.

Word 2 is the address of the diagnostic library. This word points to the message that is displayed in case of error. If no error has occurred, word 2 is set to 77777 octal.

Verify Buffer

The Verify Buffer, which is allocated in ring 4, is a 3-word area that holds the current record processing parameters. Each bit position indicates an operating mode. Bit 1 is set for read, bit 2 for write, bit 3 for read/write, bit 4 for read/lock, and bit 5 for recover mode.

The SCANNER and PROC routines perform polling and character processing operations.

SCANNER performs the following functions:

- polls all lines in the network and upon detection of an input character, performs a series of tests to determine the legality of the input character
- calls a special-purpose routine to set pointers to four core-resident buffers:
  - Term Buffer reflects terminal status.
  - Verify Buffer maintains current record processing parameters.
  - Format buffer holds the memory address where the format is stored.
  - Data Buffer holds the input characters until they comprise a full record and require transmission.

The pointers are established when the first input character is moved into the Data Buffer. After the last input character has been verified, SCANNER calls the PROC routine.

PROC performs the following functions:

- checks the input characters against the stored format
- interprets the operation to be performed
- transfers control to the appropriate processing routine.

When the polling cycle is complete, control passes to the EXECUTIVE program. If an input/output operation is required, the EXECUTIVE performs the following:

- calls the appropriate driver
- sets the I/O request word to zero
- returns control to SCANNER to continue the polling operations.

FIGURE 1-2 Unbroken and Broken Text

2. *Text is straightforward.* These sentences are not straightforward:

   There is one major program in the application: MONITOR.

   No zeros to the right of the decimal point are suppressed.

   But these sentences are:

   MONITOR is the major program in the application.

   Zeros to the right of the decimal point are not suppressed.

3. *Text is clear and concise.* This sentence is not clear:

   If the list option is selected, the last scanned line of source text is listed.

   But this sentence is:

   If you select the list option, the last line of source text scanned by the compiler is listed on the output device.

   This sentence is probably clear:

   From the programmer's point of view, you should determine what routines are needed, what arrangement the routines should be grouped in, and when it is necessary to consider making procedure calls.

   But this sentence is clear and concise:

   Determine what routines are needed, how they should be grouped, and when procedure calls should be made.

4. *Text is precise.* Precise text is complete text, leaving nothing to the reader's imagination. The description of a square root routine, for example, would include operations for negative as well as positive numbers.

   This sentence is not precise:

   The RESULT field should be larger than the X and Y fields.

   But this sentence is:

   The RESULT field should be at least 4 character positions larger than the X and Y fields to prevent truncation of high-order digits.

5. *Text provides only one interpretation.* Each of these sentences provides two interpretations:

   The conversion routine may be called by the main program.