The Art of Technical Documentation

Katherine Haramundanis

The Art of Technical Documentation

Katherine Haramundanis

digital

Digital Press

Printed in the United States of America.

9 8 7 6 5 4 3 2 1

Order number EY-H892E-DP

Text and cover design: Sandra Calef
Production: Editorial Inc.; Kathryn S. Daniel, Production Manager
Composition: Paul C. Anagnostopoulos and M. Dean Hendrick, using ZzTEX
Copyediting: Dianne Wood
Index: Lois Oster

Quotations from the following works appear as epigraphs in this book: Blaise Pascal, *Pascal: The Provincial Letters*, translated by A. J. Krailsheimer (Penguin Classics, 1967), copyright © A. J. Krailsheimer, 1967, reprinted by permission; W. K. Heisenberg, *Physics and Philosophy: The Revolution in Modern Science*, copyright © 1958, HarperCollins Publishers, reprinted by permission; S. N. Kramer, *The Sumerians*, copyright © 1960, The University of Chicago Press, reprinted by permission; quotation on page 143 from Donald Knuth, *TEX and Metafont*, reprinted courtesy of the American Mathematical Society.

Trademarked products mentioned in this book are listed on page 267.

Views expressed in this book are those of the author, not of the publisher. Digital Equipment Corporation is not responsible for any errors that may appear in this book.

Library of Congress Cataloging-in-Publication Data

Haramundanis, Katherine, 1937-
The art of technical documentation / Katherine Haramundanis.
p. cm.
Includes bibliographical references (p. 235) and index.
ISBN 1-55558-080-7 (paper)
1. Technical writing. I. Title
T11.H28 1992 91-34993
808'.0666–dc20 CIP

Contents

Figures

Tables

About This Book

Who Is the Audience of this Book?

If you are a novice technical writer who works in the computer industry, or are considering such a position, this book is for you. Read this book from start to finish; don't skip around. This book will help you become a better writer; and you'll find references and suggestions for further reading when you want to extend your knowledge.

To get the most from this book, you should have the following skills and experience:

- be well trained in writing English
- have had exposure to computer systems
- have taken at least one course or had experience in expository writing, such as doing investigative reporting or writing material based on fact
- have taken introductory courses in computer science and a science such as geology, biology, physics, or astronomy
- understand the logical flow of ideas

What Is the Thesis of this Book?

The thesis of this book is that the practice of technical writing is not the same as that of scientific writing, that it is closer to investigative reporting. When you create technical documentation, you need to gather information rapidly and identify audience; these are tasks the scientist need not perform. Scientists are thoroughly trained in their respective fields and know their audience well—their readership is primarily their colleagues, people trained in their field.

Further, technical documentation is as much an art as a science. Its practices can border on the intuitive, and require creative thought on your part. The work you do requires creativity and problem-solving skills, and you must use your imagination to write technical documentation, though what you write won't be fanciful—it's based on fact. You apply analytical thought to dissect and understand the information you gather, but there are few rules on

how to gather information, or how to put that information together in a way that your reader will understand most readily.

When you gather information, you learn to work with your technical resources, and you learn from your reviewers. When you prepare your documents, your appreciation of your readers, what they know, and their reasons for using the documentation come into play. This is where new techniques are still being developed for the technical documentor.

Your art consists of the techniques you master to gather, understand, and distill your technical information; you show your craft in the effectiveness of your documentation and in your proficiency with your language and your tools.

The field of technical documentation is more than just its day-to-day practice. Its industrial practitioners are paralleled by a body of scholars in academia who help to explain and enhance professional practice. I cite academic references in text, at the end of each chapter, and in the Select Bibliography. You'll find short titles and (author, date) in the references in text or at the end of each chapter; the bibliography is alphabetical by author.

What Is the Structure of this Book?

This book is my view of what it takes to produce effective technical documentation. This is not a style guide that deals with all aspects of typography and copyediting, but presents for your use the distilled knowledge of my experience. After preliminaries, you'll find general precepts, then three chapters that address practice and techniques. The last major chapters summarize and compare software and hardware tools you are likely to encounter. Appendices gather reference material. In more detail:

- Chapter 1 defines technical documentation, and describes quality in technical documentation.
- Chapter 2 describes career paths and documentation management styles.
- Chapter 3 describes the precepts of technical documentation, and provides examples of applying those precepts. This chapter also provides background information on CALS, a strategy for moving from paper to electronic media.
- Chapter 4 describes the practices for gathering information, understanding what you have gathered, and methods for testing documentation.
- Chapter 5 describes the use and preparation of graphics, important components of most technical documentation.
- Chapter 6 describes considerations of information representation, to provide insights on how different representations affect reader perception of your documents.
- Chapter 7 describes some currently available tools and compares popular documentation tool methodologies. These tools exemplify current tools—new tools will evolve from them.

- Chapter 8 describes representative hardware systems used in preparing technical documentation. New hardware for documentation also evolves, and this chapter provides only a snapshot of some current systems.
- Chapter 9 presents a brief conclusion.
- Appendix A lists and briefly describes professional societies, conferences, and journals relevant to the work you do.
- Appendix B contains lists and tables of relevant standards.
- Appendix C shows a timeline of milestones in the development of writing and writing tools.
- The book ends with a glossary of terms, an annotated bibliography of books and papers, and an index.

A Word about Style and Conventions

The writing style of this book is deliberately gender-neutral. The style favors non-gender-specific words such as author, writer, engineer, programmer, or developer, and avoids words like he or she.

You will also find a few professional tips, indicated by the word "TIP" and an arrow offset at the side of the page.

Acknowledgments

People and libraries have been essential to the success of this document. Those who have helped with technical information include Doug Borsum, Dave Eklund, Jim Flemming, John Francini, Marty Friedman, Richard Hansen, William A. Hunzeker, Susan Hunziker, Dick Howard, John Hughes, Scott Jeffery, Steve Jensen, Rob Limbert, Sarah Masella, Dan Murphy, Chuck Murray, Robert Pariseau, Carol Perlman-Ton, Holly Platnick, Andy Puchrik, Dick Rubinstein, Tara Scanlon, Ben Shneiderman of the University of Maryland, Nicole Yankelovich of Brown University, and Jan Walker of Digital's Cambridge Research Laboratory.

Reviewers who have spent long hours commenting on drafts of this book include John Herrmann, Dick Howard, Tara Scanlon, Dr. Peter Jordan of Tennessee State University, Dr. Robert Krull of Rensselaer Polytechnic Institute, Molly Miller of Ascend Communications, Inc., Linda Pesante of Carnegie Mellon University, Carole Yee of New Mexico Tech, and several anonymous reviewers (you know who you are!). To these dedicated professionals I owe my sincerest thanks.

I am also grateful to my management, Susan Porada, Kathy Richer, Susan Gault, and William Keating, for their support in this project allowing me to use Digital Equipment Corporation computer facilities to create my drafts and graphics.

The libraries of Digital Equipment Corporation, Northeastern University, J. V. Fletcher Library, Westford, Mass., Chelmsford Public Library, the Stevens

Memorial Library, North Andover, Mass., and Boston College have been most generous in lending books.

I also acknowledge with sincere thanks the support and wise counsel of my editor, George Horesta.

The opinions expressed in this book are mine, and do not necessarily reflect the views of Digital Equipment Corporation. I have benefited from the advice and comments of my technical resources, but any errors remaining in this book are of course mine.

The Art of Technical Documentation

This page intentionally left blank

Technical Documentation Defined 1

Things are always at their best in their beginning.

Blaise Pascal, *Pascal: The Provincial Letters*, No. 4, 1656/7

When you write technical documentation, you follow a discipline and create specialized types of material. The techniques you learn to use are generic; you will find that you can use these techniques whether you are developing documentation for computer hardware or computer software. You can also use these same techniques when you create technical documentation in other industries and for other business environments.

This chapter describes what technical documentation is and what constitutes technical documentation. Technical documentation is both the work you do when you prepare technical documents and the result of your work. This double meaning for the phrase is like the double meaning of the word "painting": both the work the artist performs, painting, and the result of the artist's work, the painting.

In the context of this book, *technical documentation* is any written material about computers that is based on fact, organized on scientific and logical principles, and written for a specific purpose. When you write technical documents about computers, the subject you write about has a technical nature and you write with a specific purpose. The scientific and logical principles you follow are:

- To substantiate, or be able to substantiate, the statements you write
- To develop your ideas logically

This is a narrower definition than that of technical writing, whose definition is still developing. Some suggest, for example, that technical writing is writing for a purpose, while others suggest that it is a language a social group has agreed is useful.

All technical documentation is nonfiction (though sometimes you may feel you are writing fiction!), and all technical documentation has technical content—whether the purpose of the piece is reportage, instruction, or persuasion.

According to *Webster's Ninth New Collegiate Dictionary*, the term nonfiction appeared only in 1909. Technical documentation is even newer. Newspaper

articles, magazine articles, and biographies, for example, being based on fact, are all nonfiction, but such literature is not technical documentation. However, a newspaper article can be technical documentation if the article describes a technical subject related to computers, and if the writer handles the subject without exaggeration or gross inaccuracy.

In some engineering organizations, "documentation" includes the parts lists for a product or the engineering drawings or specifications prepared by engineers, but you won't usually work on this type of document, except perhaps as a technical editor. This book primarily addresses the writer creating original technical documents rather than the editor of documents written by someone else.

Types of Technical Documents

There are three types of technical documentation: marketing materials, materials that report, and instructional materials.

Marketing or sales pieces are intended to convince or persuade; pieces that report state the facts without a persuasive or instructional slant; and instructional pieces can both instruct and train. Pieces that instruct include traditional documents that describe a product for the user. Sometimes you may have opportunities to provide materials for use in presentations.

Table 1.1 shows the documents you may write in these three categories. This book primarily addresses writing instructional materials in the computer industry.

Now that you have a view of what technical documentation is, you need a perspective of what makes high quality in technical documentation. This is the subject of the next section.

Quality in Technical Documentation

High-quality technical documentation is:

- Accurate
- Complete
- Usable
- Clearly written
- Readable
- Logically presented

Table 1.1
Technical Documentation Types

Marketing	Reporting	Instructing
brochure	magazine article	customer manual
case study	newspaper article	user manual
sales pamphlet	journal article	instruction manual
press release	internal publication	site preparation manual
product handbook	technical paper	installation manual
product catalog	progress report	owner's manual
marketing script	internal report	reference manual
marketing talk	annual report	maintenance manual
sales presentation	blueprint	system manager's manual
technical summary		operator's manual
advertising copy		technical description
white paper		functional specification
"mock" paper		user interface specification
testimonial		glossary
data sheet		training manual
product brief		quick-start guide
application guide		presentations
		course materials

- Concise
- Written with appropriate language
- Grammatical
- Appropriate in content and scope
- Presented in an appealing package

Accuracy

An accurate document contains neither errors of fact nor misstatements that will confuse the reader. For example, when you write that to perform a task the user should press the E key, and the user really needs to press the CONTROL and E keys simultaneously, you make an error. Or you might omit a step in a procedure or add an extra space in a command line. You can commit these errors if you write your document hurriedly or don't become familiar with your product. You should also verify, or have someone else verify, any procedures you write.

TIP ➧ ***Take the time to verify your facts.***

A good way to verify a procedure is to draw a flowchart or a Nassi-Shneiderman diagram (see Chapter 4) of the procedure. This can often show missing steps or steps that lead nowhere. Another effective way to verify a procedure is to have someone who does not know the procedure follow your written text and perform the procedure. The tester can mark up your written text, or you can watch the tester follow the procedure and note any difficulties the tester has. (For more on these techniques, see Chapter 4.)

Completeness

A complete document does not leave out something that is important to the reader. For example, if you write a reference manual, be sure that it contains all the commands or statements of the product.

If you write a procedure, be sure there are no missing steps. Or if you write a manual to describe the error messages the user can see on a system, make sure it contains all the error messages. If you leave even one out, and the user sees that one on the system, confidence in the completeness and accuracy of your documentation will be eroded. So check your document carefully to be sure you have found and included all possible error messages. Your technical resources must help with this task by providing you with a complete list of error messages, but you are the one who must verify the completeness of your document.

Usability

A usable document is one your reader can use easily—it is not too bulky or designed in such a way that your reader must work extra hard to find the information. Usability applies both to printed books and online texts. The organization of your information is important too. If you don't organize your information so that your reader can grasp the information quickly, you will only frustrate your reader. Your reviewers can help you find out if your document is usable (more on these topics in Chapter 4).

Usability tests help you determine if you have created a usable document and show you how to correct faults in documents that are less than usable. Usability tests also help you analyze the effectiveness of the information you provide.

When people began to write technical documentation, they practiced their art intuitively—the work was essentially a craft. As we have gained more experience, we have learned more about the effectiveness of technical communication and have developed analytical methods to examine and test documents.

Usability tests, properly applied, can be extremely effective. If you conduct such tests and modify your information accordingly, you will find that your documents and information packages become more effective.

You can conduct your own usability tests—you don't need an elaborate laboratory setup to do this testing (more on this in Chapter 4).

Clarity

Write the text in your document clearly. Follow the rules of good writing, and trust your reviewers to help you find those muddy passages or that flawed logic. Even a reviewer who just puts a question mark in the margin of your draft helps you improve the clarity of your document. If that reviewer doesn't understand what you wrote, others won't either. Take the opportunity to discuss the information in the confusing paragraph or sentence with your reviewer. You will very likely find there is another way to present the information that is more clear.

The prize for ambiguity in writing belongs to the order issued by British headquarters at the battle of Balaclava (1854): "to advance rapidly to the front and try to prevent the enemy carrying away the guns." The intent of the order was for the Light Brigade cavalry unit to retake guns that had just been captured by the enemy; the result was that the Light Brigade charged an entrenched enemy position in the opposite direction and was slaughtered.

Most of what you write won't have such dire consequences, but if, for example, you are describing a software application that controls a nuclear power station, you might find that what you write is a critical piece of documentation.

So when you are writing technical documentation, be aware of the ambiguity of what you write, and examine your work to eliminate ambiguity wherever possible.

Readability

Your text and document must be readable. What does this mean? If your document is readable, your reader will understand it. For example, if you are writing for experienced programmers, you can expect they will be familiar with the technical terms of the trade. But if you are writing for novices, you need to explain all your terms and be particularly careful in consistent use of those terms. Otherwise you will confuse your readers. Readers don't expect synonyms in technical documentation; in fact, they will be confused by an alternative word and may wonder if you are introducing a new concept.

If you like, you can use a readability test software program to help you determine if your prose is written at the level of education expected of your reader. You'll find more about readability in Chapters 6 and 7.

Logical Progression

Your text must flow in a logical progression, from simple to more complex or from start to finish of a procedure. To some extent, writing logically is part of writing clearly, but logical progression should be evident in your organization of information and in how you approach your subject.

Conciseness

Avoid verbiage and keep your sentences and words as short as possible. Learn to discard ruthlessly words that add no information to your text. For example, avoid phrases such as "in order to" ("to" does the job). You can find these extraneous words by actively reading what you have written. Also read what you have written after an interval—some read their words aloud. Always use a short word rather than a long one if the two words have the same meaning. For example, don't use "utilize" when "use" will do. Many style books contain lists of such short substitutes for long and pompous words and phrases.

When you must get information across instantly, use the *one-page display*: distill critical information to a single page. You will need to put a lot of thought into a one-page summary of anything complex, but such a piece can be extremely effective. You will often need to use a diagram or a table to compress information onto a single page. Command lists, balance sheets, and reading paths are examples of one-page representations of critical information. For an example of a one-page8 display, see Figure 4.5.

Appropriateness of Language

Establish the language that is appropriate for your intended reader. When you write a sales brochure, for example, you have perhaps twenty seconds to catch the reader's interest: your text must be brief. So using the right words and the ideal turn of phrase is critical. The shorter your piece, the more important each word it contains. A sales piece will use at least one of the Great Attractors of technical documentation (for more on the Great Attractors, see Chapter 6).

Most technical documentation is written in a rather formal style. For example, when you write a computer manual, you will avoid slang and contractions. This book, although about technical documentation, is less formal than much technical documentation—it is not a computer manual, so I take some liberties with my writing style.

A good way to find out what language style is best for a specific piece is to read other pieces written for the same reader. That helps make you familiar with the terminology and phraseology of the subject about which you will

write. Of course, if you are developing a piece for a wholly new readership, you have to rely on yourself and your reviewers to ensure that you use the right mix of words, tone, and phrases. You may be able to examine competitive literature for ideas about appropriate writing style.

You will find examples of several writing styles in this book.

Grammaticality

Be sure all your sentences and phrases are grammatical. Readers will give up if you make too many grammatical errors. These errors include errors in spelling and punctuation as well as errors in grammatical form.

For example, the sentence "The motor shut down when you press the disable key" is ungrammatical because the subject ("motor") and the verb ("shut") are not in agreement. (The verb should be "shuts.") Ask a copy editor or literary editor, or perhaps a writing peer, to review your work. Their constructive criticism can be invaluable in helping you find and correct such errors. Spell-checking software can find some spelling errors, but it won't warn you about grammatical errors. If you are not sure about your grammar, enlist the aid of another writer or an editor, or consult a grammar book. (You will find the names of some good grammar books at the end of Chapter 3.)

Appropriateness of Content and Scope

Your piece must have appropriate content and scope. For example, there is no point writing a piece for the novice that contains all the details only an expert could want to know, and there is no point writing a step-by-step user manual for an expert who will find the progression of thought and exposition much too slow.

You can verify your content and scope by checking your table of contents against the norms for your readers, by contact with your readers or potential readers, and from your technical resources.

Appeal of Package

The excellent book you have written won't be read if the packaging that presents it to the reader is awkward, messy, or hard to use. If possible, work with those who guide the printing process to ensure that your document resides in an attractive package when it reaches your customer.

Part of packaging is binding, which is discussed in Chapter 6. Consider the convenience of your reader, the lifetime of your book, and how frequently you will need to update your book when you think about packaging.

Further Reading

You can explore the quality elements of technical documentation by reading *The Elements of Style*, third edition (Strunk and White, 1979) and a good style guide (you will find several listed at the end of Chapter 3). You can also gain an appreciation for a variety of styles in English prose by reading *The Reader over Your Shoulder* (Graves and Hodge, 1964) or *Language in Action* or *Language in Thought and Action* (Hayakawa, 1941, 1978).

For More Ideas

Some popular books and articles on scientific or technical subjects can also help you understand the subtle elements of high-quality writing you can use in your work. Examples include *The New Physics* (Taylor, 1972), articles in *Science News* and *Scientific American*, and the works of Martin Gardner. *Popular Mechanics* is another good source for clearly written technical articles.

A Career in Technical Documentation 2

A scribe whose hand moves as fast as the mouth, that's a scribe for you!

Sumerian proverb, c. 2400 B.C., translated by Edmund Gordon,
cited by S. N. Kramer in *The Sumerians*.

To give you an idea of what technical writers do, these next paragraphs provide scenes of the kind of day many writers have. The names and projects are purely fictitious.

Larry, the Documentation Project Leader

Larry Leader has been a writer at X Corporation for eight years. In that time, he has worked on four different software products and is now documentation project leader for the Xproduct, a software system that runs with the ABC operating system. Larry's job as project leader is to coordinate the work of the other writers, ensure that they keep up with software changes, and be a resource to them.

Larry starts his day at 8:05 A.M. when he drives into the large X facility along Route 495 in Marlboro, Massachusetts. His well-lit office on the third floor contains color-coordinated office furniture, a terminal, and a work station system. His neighbors include developers working on the Xproduct software as well as other writers working on the project.

Larry sits down and taps the Return key on his work station; the screen comes to life, displaying a password window. After entering his password, he opens a window to read his mail and then opens another to make changes (edits) to a document he needs for the writing team meeting he will run, scheduled for ten o'clock.

By 9:55 Larry has completed his strategy document and run off ten copies for his meeting. He picks up his project notebook and walks down the aisle to the conference room he has reserved. A writer is there already, and they chat amiably as the other writers enter the room in twos and threes.

TIP ***Write out your agenda for all to see.***

Larry writes his agenda for the meeting on the whiteboard: current status of Version 1.0, plans for Version 1.1, documentation strategy for Version 1.1, and

assignments. He ends his list with "other" to encourage discussion of topics not covered previously. By 10:05 he starts his meeting.

Jamie Junior, a novice writer who has been on the project for about eight months and who is new to X, asks when she needs to give her last book, the installer's guide, to production, the group that will prepare camera-ready copy. Because the software was not done when expected, the final software date has been delayed by two weeks. Jamie asks if she should keep her book open for the two-week period. After some discussion, and after hearing from several other writers on the project, Larry advises her to complete her document as planned but to check with him when she plans to hand her files to production.

Annie Able, writer of the programmer's reference manual, also asks about the changed date. Will it affect her book? After more discussion, Annie agrees to complete her book on the date planned.

Larry passes out the new version of the documentation strategy, points out the changes he has made, and suggests tentative writing assignments. Jamie will update three small documents for the next version and agrees to write the Release Notes. This is a specially challenging assignment because she must work very closely with development during the last few weeks of the project and keep up with a steady stream of changes during field test.

Annie agrees to maintain the programmer's manual and to create a new reference card. The other writers agree to their assignments of administrator's manual, user's manual, help text, and troubleshooting manual. Larry assigns the installation guide to himself, in addition to a second book, the error message manual.

Larry concludes his meeting and returns to his office, where he makes his final edits to the documentation strategy and copies the completed file to the public area on the large computer system. (The public area contains files that others in the company can copy and print at their local sites by accessing the files over a network.)

TIP ▸ ***Use common directories and electronic mail.***

He then sends electronic mail to the writers, the development team, the product manager, the customer services people, and those who have expressed an interest in seeing the documentation strategy, informing them that the strategy is available; he asks them to return comments within two weeks.

After lunch, Larry attends a development meeting. The room is already nearly full with developers, qualification engineers, testing people, and writers. Desultory discussions about equipment issues cease when the development project leader arrives, carrying two small computer tapes. He holds one of them up and asks: "I have this morning's software version here on this tape; is there any reason I should not send it to our field test sites?"

This question prompts a lively discussion, with comments, questions, and

sometimes complaints from both developers and testers. "There's too little time given to full testing," someone states. Yet there is agreement that they cannot test forever—somehow they must reach the moment when they can ship a reliable, tested product. Annie Able expresses a concern about the magnitude of the changes—they seem large. One command qualifier, for example, is totally new for this version of the software.

TIP ▸ ***Keep your technical resources on tap.***

Two developers agree to help her verify how the qualifier has changed the system response so she can change the manual that day.

The meeting concludes at 3:35. As Larry returns to his office, a developer stops by to ask when the Release Notes will be done. Someone from the training group arrives to ask for fifteen copies of the manuals for a training course to be given the following week. Larry tells her about the public area, where she can get a copy of any book at any time.

Larry gets several telephone calls, and more electronic mail arrives. Dealing with these interruptions occupies him until the end of the day.

By 5:15 Larry logs off his system for the day and leaves the building; the parking lot is still more than half full. At 8:30 P.M. Larry dials into his system from home to answer some mail he did not have time to answer at his office; by 9:15 he completes his work for the day, logging off the system one final time.

Jamie, a Junior Writer

Jamie Junior, less than a year with the company, is enthusiastic about her position with X. Her first job was to update the installation manual for the software, which had been written by another writer. She arrives at her cubicle at 7:30 A.M., greeting two nearby developers who have been in since midnight to use machines in the lab. She starts her work station, checks her mail, and begins to edit a section of her manual. A couple of system prompts have changed, and she wants to improve a procedure that a couple of her reviewers found difficult to understand. The reviewers' comments helped her see that some steps in the procedure were not clear.

Soon the project consulting engineer stops by to ask her to come around to his office at nine to discuss his review comments. Before she can leave for this conference, the development project leader drops in to discuss how to get his last minute changes to her. By the time they have concluded their discussion, she needs to get ready to meet with the consulting engineer.

TIP ▸ ***Keep a notebook handy.***

Gathering up her notebook and a clean draft of her manual, Jamie finds the consulting engineer in his office. They find a small conference room

and go through his comments. He has made marks on nearly every page and for each mark provides a long, verbal explanation. Jamie takes notes and asks questions to clarify his points. Sometimes he draws diagrams on the board, sometimes he goes over completely new information, and occasionally he mentions information that belongs in another book in the set. (Jamie's book is only one of several books the customer receives with the product.) They conclude their discussion at 9:55 so Jamie can go to the documentation project meeting.

After the documentation meeting and a light lunch, she returns to her office at 12:50 and begins to rework her manual based on the morning's discussion. When she finds she cannot explain a new concept or procedure clearly, she seeks out the consulting engineer, but discovers he is in an all-afternoon meeting. She finds the development project leader, who clarifies several points for her. "You mean you have to put the tape in first, then put it in a second time?" she asks. "Yes, that's the way we need to do it for now," he explains. "But we're fixing that. When we've changed the software, once will be enough."

TIP ***Have all your graphics in place for your final draft.***

Somewhat frustrated by this and similar exchanges, she returns to her cubicle and makes a note of the current status of the procedure and the probable changes. At four she goes to meet with the illustrator who is creating the diagrams she needs in her installation guide. They discuss the last-minute changes that have come up, and the illustrator agrees to have new drawings done electronically by the following Monday, when Jamie will need them for her final draft review. At 4:50 Jamie leaves the building after rather a trying day.

Example of a Project Team

Marta, the Documentation Supervisor

Marta Manager has been with the company almost ten years, after five years with another computer company. She was a writer and project leader before moving into management. Now she staffs projects and guides the writers who work on the projects that her group supports. She arrives at her office at 8:20 A.M. and answers electronic mail and phone messages until it is time to attend a morning project meeting. A writer stops by to ask about taking a course in C programming as she is leaving for the meeting. At the conference room, she notes the new faces in the project team. The only people she knows are the development supervisor and a development project leader.

The meeting opens with introductions, and Marta learns the names and functions of the people sitting around the spacious table: Mary is the product manager, who works closely with customers and the sales and sales support groups; Hank represents the service group who will support the product in the field; Terence is the hardware engineer who helped create the hardware part of the product; and Simon is the training contact. Arlene, the development project leader, mentions that the manufacturing person, Darlene, and the field test coordinator, Joan, could not be present for this meeting. After introductions, Arlene launches immediately into product scope and proposed schedule. Team members then provide status of plans and deliverables for the project.

They conclude the meeting at 11:25, and Arlene says that minutes will be sent to everyone in the next few days.

Returning to her office, Marta finds more electronic mail messages on her screen and two drafts of manuals on her chair. On her way to lunch, she takes a manual she had reviewed at home the previous evening to the writer. After a quick lunch, she answers her critical mail and then starts to draft a report on a customer visit from notes she made during the trip.

A few moments later, Arlene drops in to ask about documentation plans and requirements. "This seems like a product for the system manager," she says, and Marta agrees. "If that's the case, then we should try to do a system management documentation kit and context-sensitive help," she suggests. Arlene responds, "We're not planning context-sensitive help—there's no time in the schedule." "That doesn't sound like the right reason not to do it, when customers expect it nowadays," Marta protests. "It will mean more documentation, you realize," she continues. Arlene grins and says, "Maybe it's easier to do it that way."

Marta promises a rough estimate of the number of writers needed to do this work for the following week. She then returns to her report. A writer stops by to ask about taking a seminar. She describes the content and why it will be valuable to her, and Marta agrees that she can attend.

A little while later, Jamie, who reports to Marta, arrives, and they review the status of her project. Jamie describes where the development team is in implementing the software and getting to field test. She is anxious about her role on the project. The software is complex, and she is the sole author of the manual that describes the system installation. She wrote a fine plan for the document and an excellent outline, but now cannot get all the information she needs from the developers about software details. What does this command do? When should the user follow that procedure? The current draft is also weak on the basic concepts of the software.

As they discuss the problems Jamie is having, Marta suggests that it may be time for a documentation review meeting.

TIP ***A good time for a review meeting is when there are unanswered questions.***

The meeting will give Jamie the chance to obtain direct feedback from the developers as a group. Conflicting review comments can be resolved, and she can ask questions about the basic concepts to get views from several people. Marta recommends that Jamie add as many illustrations as possible to the draft she produces for the meeting, and they decide on a meeting date.

When Jamie has gone, Marta completes the draft of her report. She concludes her day in the office by updating her slides for a presentation she is scheduled to give the following morning.

By 6:05 P.M. she leaves her office, carrying her project notebook and transparencies for the following morning, and a draft manual to review.

Consuela, a Consulting Writer

The time is 10 P.M. The office on the second floor is dark. Suddenly the process lamp on the printer in the corner lights up and the printer starts to print pages. Consuela is printing her document due the next morning. When the printer has done its work, it shuts down. The draft remains in the printer until it is picked up the following morning at eight.

Consuela Consultant has worked as a writer for many years, often in a consulting capacity with different corporations. She recently developed several technical brochures for the company and has now written a specialty handbook for marketing a new product. While familiar with several authoring tools such as Microsoft Word and WordPerfect, she has agreed to use locally available tools to create copy for the marketing group. Once she has created her copy and it has been reviewed by the technical experts, someone else will take over preparing the final pages containing both text and photographs.

When she arrives at the facility at ten the following morning, she goes directly to the cubicle of the contracting office where she had queued her file to print the night before. The contracting staff have already looked over her draft and discuss with her when she should return and any changes they see needed eventually. They agree on a date three weeks later, to give time for the draft to be circulated and reviewed. She offers to run a meeting where reviewers will get together to discuss the draft and immediately resolve differences and conflicts. By 11:30 she leaves the building, off on another assignment at another company.

Writing Tasks

From these fictitious examples, you can see that writers perform many tasks and are often full members of development teams. Writers are advocates for the user, customer, and reader. Your focus as a writer is different from the focus of your developers, who concentrate on designing and writing software.

Thus you are a member of your development team and provide perspective and insight that help improve customer use.

Career Growth in Technical Documentation

You may arrive at your new workplace armed with a degree in technical or professional writing, or you may be making a transition from another related field such as programming or science. Perhaps you studied humanities and then became interested in computers.

When you start on the first rung of a career in technical documentation, you can expect to work with other writers and perform fairly simple tasks. In some companies, you will write data sheets or information sheets. Sometimes you will update a document that has been written by another writer. When you have more writing tasks and experience behind you, you will begin to write your own documents—planning them from the beginning, creating drafts, working with your technical resources, fielding comments from your reviewers, and sometimes negotiating with your technical resources to obtain substantive comments on your drafts.

Further along in your career, you will probably manage or coordinate the activities for a multiple-writer project. In some companies, this job is called lead writer or project leader. If you become successful at leading projects of moderate scope and technical complexity, in a corporation that is large enough to require a higher level of supervision, you may move into documentation management.

You may decide to move into documentation or writing management after you reach a plateau in your writing development, perhaps after you have written many different technical documents on a broad range of subjects.

Take opportunities to broaden the scope of your portfolio by writing marketing literature, traditional documentation, and online texts. As you continue to grow in your career, you will increase both the depth and breadth of your experience. To increase your depth, concentrate in a single area and explore all its aspects. For example, for one software product, write for every level of expertise, from novice to highly experienced programmer, system manager, or database administrator. To increase the breadth of your experience, increase the diversity of products you write about and expand the types of documents you write.

At your first job, with all the appropriate credentials, you will probably receive a starting salary in the mid-twenties, perhaps around $24,000 to $30,000 per year. After some years of experience, your salary will grow into the thirties, and if you have been in the field for ten or more years, depending on

your skills and market opportunity, you can command a salary in the forties or fifties. In a large corporation, you can perhaps find positions that pay more, such as the highest levels of consulting writers and top management posts.

Documentation Management

There are four ways computer documentation is managed in most U.S. corporations: as a singleton operation, with writers collected in a separate group, with writers integrated within the engineering community, or with outside consultants. The style where you work depends on the personality or culture of the company and on the company's management practices.

The Singleton Operation

In many small corporations, and sometimes in small, specialized divisions of larger companies, you may work as a singleton operation. You will be a lone writer working with a few others, your software or hardware technical expert, perhaps a company salesperson, and maybe even the vice president who takes a personal interest in all aspects of the corporate product.

In this situation, you must do everything: you will need to learn about your products, write about them, and use some text processing tool to create the sheets you send to the printer for copying in quantity. You may do other jobs as well. For some writers, this is an extremely satisfying environment—you do your own documentation management and report to someone who knows the documentation is needed but relies on you to get it done.

The Separate Writing Group

In some corporations, the writers are a separate group and sit in an area of the company that may be part of marketing or perhaps the typing pool. They may be in a different building from the engineers or programmers. Documentation management is typically by a single manager or supervisor, who sees the writers employed by the company as tools to do specific pieces of work.

When a document is needed for a product, whether a manual or a data sheet, a request is made directly to the supervisor of the writing pool, who chooses a writer to carry out the assignment. Projects tend to be short, and a writer typically does not get involved in developing product documentation until the product is well under way. In this environment, it is difficult for a writer to help improve the product itself.

You may have your own desk and access to a word processing system, but not all writers will have all their own equipment. Perhaps a typing pool will be available to type the handwritten copy you produce.

Part of Engineering

In corporations where writers are part of the engineering or product team, they tend to have desks or cubicles within the engineering organization. The philosophy is that writers need to be close to their engineering resources, to talk with them on a daily basis and keep up with changes. The documentation managers or supervisors have offices or cubicles near their writers, but sometimes writers are dispersed over a wide geographical area.

The Outside Documentation Consultant

If you act as an independent contractor, you obtain assignments from different companies and work on each independently. You will probably travel more than the writer who works for a single company and experience a wider variety of projects. You may restrict your travel to local distances, but depending on your expertise, you may have to travel more extensively. You may have your own hardware and software tools, or you may use the facilities of the company that holds your contract.

The work of the singleton operation, the integrated environment, and the outside consultant tends to be quite unstructured, unlike the work of a separate writing group. If you work in any of these unstructured environments, you will make many of your own decisions, though you will always need to keep someone informed of what you are doing. Depending on your preferences, you may feel uncomfortable because of this lack of structure, or you may thrive on it.

In a more structured environment, you will be given assignments and deadlines, and will need to meet those deadlines. In this environment, deadlines tend to be short, and there is rarely time to plan a document or a documentation set. You may be asked to develop an outline before you begin to write a manual, but for most assignments, an outline won't be expected. Sometimes this can lead to difficulties if the book you write is not what the customer needs.

Management of any writing group involves assigning writers to engineering projects, working with the writers to ensure that project needs are met and books prepared on time, evaluating the work of the writers who report to you, and managing the resources the writers need to do their work—for example, equipment, office space, or training. Sometimes documentation management must negotiate with engineering to ensure that appropriate funds are

available to support the requests for documentation. This will depend on the internal organization of the corporation and how it runs its business.

This chapter has shown you what a career in technical documentation can be like. The next chapters describe techniques that can help you be successful in your career. The computer industry, which employs many writers, is still in its infancy and growing. The potential for considerable growth in the career of technical documentation parallels the growth of the industry.

Further Reading

The *Proceedings* of the International Technical Communication Conference (ITCC) are a good source of information on activities in the field. The *Journal of Technical Writing and Communication* is another source for topics of interest to writers and managers of writers. Surveys of salaries for writers in the field are published periodically by the Society for Technical Communication; some include information on documentation management. For suggestions on writing career opportunities, see *English for Careers* (Smith, 1985).

Precepts of Technical Documentation 3

But yet I run before my horse to market.

William Shakespeare, *Richard III*, act 1, scene 1, c. 1594

This chapter describes basic precepts that will become second nature to you, the more technical documentation you create.

You will find that your work is both linear and iterative. As a writer, you must keep in mind the overall process from research and planning to final result, and you must also have the flexibility to deal with reviewer and editor comments, make changes, and proceed through the overall process with a number of iterations. Knowing the overall process helps you focus on your work, while the iterative process helps you improve the material you prepare.

Following certain basic principles can help you ensure that your material is accurate, appropriate, complete, and timely. These include:

- Following a specific work methodology
- Researching and understanding technical concepts
- Interviewing and discussing material with technical experts
- Performing audience or reader analysis
- Evaluating and developing appropriate screen or book design
- Organizing your material
- Establishing and adhering to schedule
- Using the forms of writing appropriate to your subject and reader
- Reexamining and testing material
- Performing required editorial tasks

The sections that follow describe these activities in more detail, and Chapter 4 expands on the techniques you can use.

When you write technical documentation, you act as a bridge between the designer of the hardware or software product and the user, typically a customer who obtains the product. As a writer, you create part of this product—the information component. You may present information to the customer in either printed or online form.

You cannot take the design of the hardware or software component of the product and map that design into your information component. You must create the design of the structure, tone, and content of your information

package, and use your own methods to address your reader. Developing the right information structures, writing with the right tone, and providing the relevant content constitute a large part of your work.

For example, you cannot expect to read a functional or implementation specification and follow the structure of the specification in creating a manual. You will have to develop your own information design. You will also find that your design can change in some measure as you progress in the project, so it helps if you keep in mind that your process in developing an information design is iterative.

When you follow the basic principles and work methodology of technical communication, you can help a documentation project avoid common errors, such as late participation of technical documentors in a project, incorrect assumptions made about your reader, and lack of an iterative process to design, develop, refine, and finalize informational materials.

Work Methodology

Part of your *work methodology* is the development and internalization, through training and experience, of four precepts. The work methodology is a process you learn that gives structure to what you do and how you go about gathering information, finding ways to use that information, and verifying that what you do works. The four precepts, set in rhyme to help you recall them and in the order in which you will use them, are:

Know your subject
Know your reader
Know the rules
Know your tools

- "Know your subject" means that you must be knowledgeable in your subject before you can write about it, although you don't have to be an expert.
- "Know your reader" means you must have a fair idea of the reader for whom you are writing.
- "Know the rules" means that you need to be experienced in the rhetorical and grammatical aspects of English, including spelling, punctuation, and all other mechanical aspects of writing skill.
- "Know your tools" means you must become proficient in the use of the editing and text processing tools you will use to create text and graphics.

When you learn and put into practice each of these precepts, you will always be successful in documenting even the most complex products, and you will provide just the right amount of information in the most effective

form to your reader. As a writer, you make many decisions about what to convey to readers and how to convey it; how you present information can significantly affect reader comprehension.

Know Your Subject

Because you must understand your subject before you can write about it, the first precept requires that you do your homework. Knowing the subject gives you the basis for what you write and makes it possible for you to ask relevant questions of your technical resources and understand their responses. Most technical resources won't want to answer your questions if you haven't made an effort to understand the subject.

Sometimes you may have very little time to get to know your subject, but most writers of technical documentation love to learn about new products and new tools and so learn rapidly.

Technical Concepts

Under the precept "know your subject," you must learn the basic concepts of the technical field in which you work. You can only develop accurate and appropriate material if your level of expertise is close to that of your readers. In some cases, you must know more than your projected readers to develop effective materials for them.

If your level of knowledge is too far above that of your readers, you might create material that is incomprehensible to them because you make too many assumptions about what they know. If your level of knowledge is too far below that of your readers, you may make errors of omission, leaving out important concepts or describing concepts insufficiently.

In the computer industry, there are several levels of expertise in both software and hardware concepts and methodologies. You need not be an engineer, programmer, or software developer to write material that is accurate and targeted correctly to your readers, but you must have sufficient knowledge to clearly understand the training level of both the engineer or software developer and of your readers. You must at least understand the basic vocabulary of the field and the general principles of hardware or software operation.

Use of Technical Experts

Much of what you do is dealing with, interviewing, interrogating, and working with experts who are knowledgeable in a specific area. Only by experience will

you develop the techniques to learn your subject at the appropriate level of detail, to ask the right questions, and to immediately comprehend the answers given by the experts with whom you work.

You may be readily accepted by the development or engineering community and thus gain quick access to technical experts, or you may face significant frustrations in obtaining the experts' time and attention. Earning the respect of your team peers requires study, dedication, good interpersonal skills, sincere interest in the subject about which you are writing, and respect for those with whom you work.

Writers develop many strategies to obtain information. The most effective writers are generally those who can meet with their technical resources as peers and who can suggest improvements to the hardware or software as well as accept comments and recommendations on the drafts they present to the experts for review.

Sometimes you will have to be creative in finding people who can provide technical information. At other times, you will be flooded with information you must digest before you can produce the needed document. You will find that your technical experts are most receptive when you are consistent about meeting your schedules, meticulous with technical details they provide, and diligent in trying to understand the information they convey. When you and your expert work together as a team to create a document, your reader will always benefit.

Know Your Reader

Under the precept "know your reader," you must identify the readers to whom you are writing, evaluate what they require of the documentation, and determine why they will use the documentation. Audience or reader analysis is an area still undergoing considerable research, and no absolute rules are yet established that enable you to unequivocally identify and define the readers to whom you write.

You can classify readers by three main characteristics:

- Education
- Experience
- Expectations and needs

In the list below, levels within these three characteristics map into specific reader categories. For example, the novice has no computer experience and needs to perform only a few tasks, but the system programmer has both traditional and technical education, significant experience with computers, and stringent requirements for plentiful, accurate information.

As the computer industry changes, product readership also changes, and while the lists below are appropriate today, the increased availability of inexpensive systems and personal computers can sometimes blur these distinctions. For example, with some systems, the user is novice, operator, and system manager all at the same time. If you are documenting this kind of system, you won't have three different readers but only one.

For computer software, typical reader levels (in order of increasing complexity) are:

- Novice
- Experienced user
- Operator (in computer room)
- System manager
- Information systems manager
- Language programmer
- System programmer
- Applications programmer
- Diagnostics programmer
- Software designer
- Software implementor/developer
- Sales representative
- Sales support representative
- Technical service representative
- Software consultant
- Database consultant

You may never write documents for all these readers, but you are likely to write for the novice, the experienced user, the system manager or administrator, and the programmer.

In the hardware engineering realm, levels of expertise include:

- Owner
- Technician/board swapper
- Experienced technician
- Hardware engineer
- Mechanical design engineer
- Logic design engineer
- Board layout engineer
- Diagnostics engineer
- Microcode engineer
- Manufacturing engineer

You are most likely to write manuals for the owner and the technician.

Your product and the market where the product is sold help dictate your projected readership. Depending on your work environment, you may have

direct access to existing customers or you may work through another person or another group to establish a customer profile. Understanding the customers and what they do with the product is an important aspect of your customer analysis.

If you work in a corporation that creates a specific type of product, for example, a spreadsheet application, the experience of your company with its existing customers and what they expect of the product will be invaluable in understanding your customers. You can use the documentation of your existing products as a model.

If you are working on a wholly new product, with no previous models to follow, first determine approximate reader levels based on the intended market, then refine those estimates by asking potential customers why they would use the product. That will help you decide what must go into the documentation. If there is insufficient time to do this, you must rely on the recommendations of those who understand the business need the product is intended to meet.

The following paragraphs provide guidelines on writing for typical readers. These are restricted to instructional types of technical documentation and don't address promotional literature written for marketing purposes or material written only as reports. You'll find descriptions of reader types first, with information on what kind of material they want. In the next major section of this chapter ("Know the Rules"), you will find sample outlines of books written for these readers.

Writing for the Novice

When you write for the *novice*, develop ideas simply, write in a tutorial fashion, and provide clear instructions.

TIP ***Try to make no assumptions.***

Try not to assume that your readers have any experience with the system. This can be surprisingly difficult to do. Your tone must not intimidate your readers, and even the look of your document should be friendly. Don't write a 500-page book for your novice—this will seem so intimidating that your reader may not even open it. Keep your books for the novice as short and clear as possible. Once a novice has completed your book or tutorial, the next instructional manual won't seem so daunting. By all means encourage your novice readers as much as possible.

TIP ***Keep your examples short.***

Provide working examples, but keep them exceptionally short, and be absolutely sure they all work. There is nothing more frustrating to the novice,

and more demoralizing, than to use an example from a book and find that it doesn't work. An expert may be able to realize that the example itself was wrong, but a novice won't and may lose a lot of time. Keep your topics short, and be very selective in the topics you cover. Don't try to give too much information. In writing for the novice, "less is more" works best.

Writing for the Experienced User

An experienced user needs less tutorial information for usual tasks and often needs more detail for some common tasks than the novice does. *Experienced users* may want a "getting started" document that brings them quickly up to speed in the concepts and use of a product. Such a document is more common for computer software than for hardware.

Often experienced users will be familiar with a similar product; thus what they need is not a tutorial for a novice but a conceptual and easily used document that can get them usefully working with at least the most prominent features of the software within half a day. A document of this type can be quite a challenge to write.

TIP ***Your examples must work.***

The experienced user always wants working examples, but sometimes it is quite difficult to develop a set of working examples that illustrate the features of the product. Depending on the complexity of the product, you may be able to develop your own examples or you may be able to have good, short examples provided by the programmers or developers who are creating the software product.

When you ask programmers to provide working examples or sample applications, however, be aware that they may not always realize the need to keep examples in documentation short and explanatory. The examples you provide must not only show how the software works but also be short enough to be easily understood. If, for instance, you have a working example that requires 50 pages of software code, it will be much too complicated for you to explain to your readers in any reasonable amount of time, and it will probably take your readers weeks of study to comprehend. So even for a complex product, keep your examples simple.

You can expect your experienced user to understand computer concepts and similar products, but you still need to describe your own product clearly and in logical order. How you structure your document for an experienced user depends somewhat on the product, but in general you will provide introductory or concept information and a "getting started" section, and then go directly into commands and perhaps customizing information. You will rely quite a bit on the knowledge your reader has already.

Writing for the Computer Operator

When you write for an *operator*, you may address readers who are either novice or experienced. If your readers are novices, provide remedial or tutorial information. If they are experienced, provide more detail for specific operator tasks than for novices.

For example, an operator in a computer room may need to know how to run certain applications or system utilities such as a batch process or disk-to-tape backup. Or the operator may need to know how to change paper in several printers or the recovery procedure for when an application fails. You must determine the typical tasks the operator performs to establish the content of any operator document.

Writing for the System Manager/Administrator

A *system manager*, also called a system administrator, requires broad concept information and modifiable task lists and procedures, and can typically use reference information directly, without intermediate, explanatory documents. The system manager has usually received more technical instruction than the operator and may even write procedures the operator carries out.

A document written for the system manager typically contains information on system security, processor operations, disk, tape, printer and terminal management, performance, system accounting, user account setup, and networking. The topics you address depend on the product you are documenting. When you document an operating system such as VMS, ULTRIX, UNIX, MS-DOS, or the Macintosh Operating System, you need to provide information on all these topics that apply to your system. Sometimes you will find ways to combine both hardware and software information in *system documentation.*

TIP ***Try to take the wide view.***

When you write documentation for an application that runs on one or several operating systems, you typically need to describe how to install the product on the operating system and anything special the system manager must do once the product is installed. For example, for a spreadsheet application used by several people who can communicate over a network, you may need to explain how the spreadsheet files can be shared or used together.

Writing for the Information Systems Manager

An *information systems manager* requires a broader view of a system, will be quite familiar with the available reference material, and will often be generating procedures for others. You provide different information for a system manager, a system administrator, a database administrator, and an information

systems manager, because the training background for these readers is similar in level but not in specific content.

The database administrator, for example, knows the details of establishing and maintaining a database, working with database users, and so on. The information systems manager knows what the products on the system are, how they are maintained, how they are installed, what the security and backup procedures are, and so on. Thus structure your documentation for technical managers according to the tasks they perform.

Writing for the Programmer

A *programmer*, software developer, or software engineer can use reference information without a great deal of tutorial or explanatory information. All different programmer types, from system programmer to designer, have considerable expertise in a specific area and can use reference material at any level of difficulty. When writing for these readers, provide basic concepts of the product and go directly to the technical information with little preamble.

TIP ***Don't digress when you write for experts.***

For experts, organize commands or utilities in alphabetical order, with no task-oriented procedures, and sometimes prepare documents like encyclopedias, with topics in alphabetical order. What programmers want is technical content directly presented. They won't object to passive voice, for example, but will be very unhappy with any technical errors (for which they will look with an eagle eye!).

For some programmer documentation—for example, computer language documentation—you may write user manuals or reference manuals or perhaps both. A user manual introduces the language concepts and explains how to use the language, how to compile, link, and execute programs in that language, perhaps how to debug programs you have written, and how to perform input and output operations with the language. Depending on the complexity of the language, you may need to create a debugger manual.

A reference manual for most languages contains descriptions of the language elements such as assignment statements or control statements, how to create subprograms or subroutines, components of statements (variables, constants, whatever is appropriate for the language), and syntax for all the statements you can write with the language. You must also cover any special aspects of the language such as recursion, and special characters.

TIP ***Show variants from standards.***

A reference manual must be a complete description of the language. If the language is supported by an ANSI standard, specify somehow (with color or

boldface type or shading) those statements in your local variant of the language that are different from the standard. (See Chapter 6 on the effective use of color.) Every computer language has slightly different attributes; your reference documentation must reflect the language you document accurately. If the user can customize the software in any way, you must supply an explanation of how to perform the customization.

For your internal programmers or developers, the ultimate documentation is the software code itself, a type of material you generally won't write, although sometimes you may help to add to or improve comments in software code or to improve error messages. Often the feedback you give to a programmer who has written error messages can be invaluable, because with your understanding of the reader, you are a good advocate for the user. You may supply working programming examples and help debug software under development.

Writing for Other Industries

In industries other than computer hardware and software, similar reader categories apply, though detailed descriptions of these categories are well beyond the scope of this book. In the automobile industry, for example, the novice reader corresponds to the prospective customer; the owner corresponds to the experienced user (someone who already knows how to drive); the owner/repair person corresponds to the computer operator (in terms of functions performed, such as adding windshield washer fluid, battery water, engine oil; changing tires; and perhaps running automated inspection or diagnostic stations); and the shop repair person corresponds to the maintenance programmer, performing the standard shop repair functions (shop repair information usually includes specialized subdivisions for electrical system, fuel system, power train, and so on).

An automobile manufacturer needs information that goes beyond what is required by the customer or shop. During automobile design, information on details of design and technologies is required to develop new concepts that can ultimately be implemented in a component or automobile. Once detailed design is done, implementation begins. This stage in the development process requires additional and different levels of implementation detail—for example, accurate layout of autobody parts and functional components. Table 3.1 compares the computer and automobile industries.

Every industry similarly separates information by its areas of expertise.

Literature developed for one reader can be used by another, but to develop the most effective literature, target a fairly narrow readership. With customer software documentation, for example, your target reader for a product

Table 3.1
Readers from Two Industries

Computer Industry	Auto Industry: Deliverable
novice	prospective customer: brochure
experienced user	experienced driver: owner's manual
computer operator	service attendant: owner's manual, folklore
programmer	shop expert: shop repair, maintenance manuals
designer	auto manufacturer: design specifications, engineering drawings, plans, tooling specifications, and so on

that runs on a personal computer can be a novice, a user with some experience, and an experienced user. Sometimes you can combine more than one reader level because the system or application you are writing about makes it easy for you to do so without flooding your novice with too many details. But in general be careful about combining your target readership.

Writing Technical Reports and Marketing Literature

Depending on where you work, you may have opportunities to write internal technical reports, marketing literature, or copy for sales brochures, advertisements, or other promotional documents. Sometimes this is the major work you do. Writing marketing literature can be very demanding because such documentation requires both an understanding of the technical features and benefits of the product and a feel for writing in a sparkling, engaging tone that attracts the reader but does not mislead. If you write a great deal of technical material for the system manager, for example, you may find it quite a challenge to shift gears and write marketing literature.

Your Technical Training

You may need technical training to understand the subject about which you are to write. Your training can be either informal or formal, and the amount of your training makes a difference when you write for different reading levels.

The best time for you to write for the novice, for example, is when you first become acquainted with a system, before you forget your initial questions and frustrations. This may also be true if you are writing a "getting started" document for a complex application such as a database system. However, if

you are documenting a complex system, you may need to have quite a lot of experience to be successful in writing a "getting started" or "quick-start" guide.

Once you attain user-level proficiency, you will need additional training to increase your proficiency level to write for the operator, system manager, or programmer. And when you have enough background to write for the experienced programmer, it will be much harder for you to write for the novice. In some situations, on-the-job training suffices, but in others, you must attend classes or seminars, take labs, study documentation, and work with others to gain the next level of skill and knowledge. When you write specialized material for the applications programmer, for example, you must be very familiar with the applications development environment.

At higher levels of expertise, writers often have many years of training and writing experience behind them. Sometimes a programmer-turned-writer may have excellent writing skills and be a key addition to a writing team; at other times, a programmer with an interest in writing may require coaching, training in writing fundamentals, or a good editor to produce high-quality documentation.

There are two separate camps on this question: do you get better documentation if you give technical training to someone who knows how to write or if you give writing training to someone who knows the technical material? My experience is that you can always provide the technical training; a person who creates technical documentation must start in the job knowing how to write. Naturally, there are always exceptions to any rule. Sometimes you find a programmer or engineer who writes beautifully, but this is rare.

The most successful technical writers have both writing skills and technical interests: they love to write and are intensely curious about computers. This combination seems to provide just the right balance in writing, interpersonal skills, and understanding technical details well enough to describe them clearly to others. Writers who lack sufficient interest in technical details may never write for the most demanding readers. As a writer, decide for yourself how much technical training you want.

Know the Rules

The next precept, "know the rules," refers to aspects of your work that are closer to guidelines than to rigid rules. These guidelines relate to:

- Organization
- Terminology
- Style

Organization of Material

An important part of your work is the organization you impose on the information you write. The organization of a document or set of documents must be appropriate for your readers, for the information the document contains, and for how you expect your readers to access the information. Organization is the single most important aspect of information delivery you consider in planning. An organization appropriate for a tutorial, for example, fails for a reference document. In English major organizational devices are:

- Topical
- Temporal
- Alphabetical
- Chunked

Topical and temporal organization can be thought of as hierarchical; when your readers understand the hierarchy you impose on your information, they can deduce where new information is likely to reside. Alphabetical and chunked information is nonhierarchical. When you produce these types of information, you must be particularly careful to provide a conceptual framework to avoid having your readers become lost in the information or frustrated because they are unable to find a piece of information.

In technical documentation, information is usually organized by topic. For example, this book addresses a single general topic, and its chapters each address groups of related topics. A good topical organization leads the reader from introductory material to topics of increasing scope and complexity.

Whether you write a system manager's manual or a programmer's manual, you must identify significant topics for your readers, then develop each to an equivalent level of detail. The significant topics for system managers are the tasks they must learn to manage the hardware and software on the system. The level of detail you provide for adding a new account should be the same as the level of detail you provide for setting up a secure system. For the programmer, keep your discussion of the language statements and attributes the same throughout an entire reference document.

TIP ▸ ***Follow the programmer learning path.***

Organize a computer language manual to follow the order in which a programmer uses the language statements. For example, in a FORTRAN manual, explain the overall structure of the program first, then the first line of the program containing its name, then how to set up language variables, how to work with COMMON, and so on. In a COBOL manual, follow a structure suggested by COBOL language statements: first the Identification Division, then the Data Division, and so on. Let the structure of a program written in the

language help you to decide on the structure of such documentation. This is how a programmer will want to learn about the language.

If you write a reference manual for the programmer who already knows the language and needs only to verify the syntax of that language on your system, alphabetical order of statements is the most common and the most useful.

TIP ▸ ***Find a common order for reference books.***

For a reference manual that provides language statements, put the statements in alphabetical order. Similarly, if you are documenting an operating system with a large number of commands, such as UNIX System V or VMS, organize the system commands in alphabetical order. With a large system containing more than about forty or fifty commands, also supply more than one way to find the command to do a task—for example, special indexes or cross-reference tables. Just because you have documented a command doesn't mean your readers will remember the command's name.

If you write procedures, place the steps in the procedure in the order in which readers perform them, and assign numbers to your steps to indicate their order. This is a temporal organization, which is common in operator manuals, installation manuals, and sometimes in system management manuals, where you describe specific tasks. In hardware, use temporal organization for hardware installation or site preparation books, owner's manuals, and user manuals.

A new method to organize information suitable for the special case, which is not topical, alphabetical, or temporal, is *chunking*, useful for online systems. When you create material as chunks of information, you decompose broad topics into small, useful pieces of information. Menu-driven systems can sometimes be described in no other way.

TIP ▸ ***Menu systems are hard to document.***

Some document authoring systems, such as the Symbolics Concordia writer system, assist you in creating this type of documentation.

In chunking you omit transition phrases and supply links or hooks between the chunks. Because you cannot know the order in which your reader will access the information, you cannot provide all the possible transition material, so with this style of writing you leave out any transitional phrases or sentences and make each chunk of information self-contained.

Format Considerations

Much technical material is created for reference only; it is not intended to be read through from start to finish. When you create such material, keep

its reference nature in mind, and clearly distinguish it from tutorial information. Format any reference material for easy reader access, and consider your reader when you design your document.

For example, establish a common format for commands in a commands reference manual and adhere to it throughout. For a language manual, establish a format for syntax and follow it consistently.

TIP ◗ ***Use large, bold heads for command formats.***

Use the command or statement as a main head, set in a larger type size than your normal text to aid reader access. Spell out your syntax conventions in the first few pages of your manual, and follow those conventions throughout. It is a good plan to include worked-out examples for each command and statement, but many reference documents do not include this level of detail. If you omit examples from your reference manual, place them in a companion user manual.

The Military Standard The U.S. Department of Defense (DOD) issued a Military Specification Standard (June 4, 1985) that defines the format and content of a variety of specifications. Called MIL-STD-490A, the standard addresses both general specifications and documents for specific types of information.

If you create military specifications that must conform to this standard, you will need to become familiar with the requirements of the standard and how to recognize and use the specialized Data Item Descriptions (DID) described in DOD-STD-2167. Data Item Descriptions include specifications for plans and documents such as a Software Test Plan (DI-MCCR-80014, AMSC No. N3590), a Software User's Manual (DI-MCCR-80019, AMSC No. N3595), an Interface Design Manual (DI-MCCR-80027, AMSC No. N3603), and a Data Base Design Document (DI-MCCR-80028, AMSC No. N3604). For more information on the content and format of these specifications, see the Military Standard, which should be available in a large public library, if not at the company where you work.

TIP ◗ ***Use CALS where required.***

Computer-aided Acquisition and Logistic Support (CALS) is a U.S. Department of Defense and industry strategy to move from a paper-intensive system to an automated and integrated computer-based operation. In a world where the technical documentation to describe a jet aircraft weighs more than the plane, and where the amount of paper required to describe a battleship weighs as much as the ship's entire load of ballast, a move from paper to electrons is inevitable.

The primary focus of CALS is improving the integration of systems that support technical information transfer for weapons systems. The goal is to have a system that can include product definitions, engineering drawings, logistic support analysis data, technical manuals, training materials, technical plans and reports, and operational data. Standards are placed between the contractor and the government to ensure that information transfer is accurate and timely, and to minimize the flow of paper.

MIL-M-28001A, developed for CALS, addresses the automated publication of technical manuals. This specification recommends use of standards for

- SGML (Standard Generalized Markup Language), the industry standard for documentation representation
- Graphics
- Data interchange

The development of other technical and data standards is coordinated by the National Institute of Standards and Technology (NIST), which works with the American National Standards Institute (ANSI). The two organizations work together to avoid overlap and use existing standards as much as possible.

If you write CALS-compliant documents using military standards, you will need to become familiar with SGML and use it when you write. Readers of your documents will most likely read them online, not on paper. Consider how you present information and how your readers will use it.

Categories of Technical Documents

The most common types of document are:

- Primers
- User guides, user manuals
- Reference manuals
- Concept or introductory documents
- Quick-start guides

Less common types of documentation include:

- Reference cards
- Wall charts or posters
- Wordless documents

The Primer

A *primer* is tutorial; it begins with simple product concepts, then leads the reader on to more complex concepts and gives a few hands-on examples. For example, a primer for the BASIC language might have the organization shown in Table 3.2.

Table 3.2
Sample Primer Outline

Chapter	Content	Rough Size
	introduction	2 pages
1	the BASIC Language	2 pages
2	functions and function evaluation	10 pages
3	more BASIC	15 pages
4	data processing	20 pages
5	solving equations	20 pages
6	extra exercises	15 pages
	glossary	6 pages
	answers	15 pages

The User Manual

What you place in a *user manual* depends on the product you are documenting. A user guide for an operating system may have the structure shown in Table 3.3.

Include in your user manual information about the documentation that comes with the system, so that once done reading the user manual, readers know what to read next.

TIP ***Create documentation maps for your readers.***

A *documentation map* or *reading path diagram* that shows in what order to read books in a large set is helpful to your readers. For an example of a documentation map, see Figure 4.5. What you select to put in your user guide depends

Table 3.3
Sample User Manual Outline

Chapter	Content	Rough Size
1	system overview	10 pages
2	basics for system users	20 pages
3	description of system tutorials, if any	20 pages
4	using an electronic editor	10 pages
5	using the mail system	10 pages
6	summary of system commands	5 pages
7	information about the file system	10 pages

on the features of the system the first-time user must know, and those things every user must know to use the system.

The Application Guide

If you are writing a user guide for an application that runs on some operating system, you may have an organization like the sample application guide outline shown in Table 3.4.

The User Guide

You may write user guides for many hardware or software components. If, for example, you are writing a user guide for a computer terminal, you may follow the organization shown in Table 3.5.

Because every application is different, has a different purpose, and may address different readers, the organization of a user manual must be flexible. Sometimes you need to avoid writing a manual altogether: for example, software that performs short tasks like running a calendar or giving you money at an electronic teller must be self-documenting. The user should never need to use any documentation other than what is seen on the terminal display. Even a reference card should not be needed for these short task applications.

But as an application becomes more complex and performs more complicated tasks, you need to provide more documentation. For example, a user guide for a sort/merge utility could contain the topics shown in Table 3.6.

Table 3.4
Sample Application Guide Outline

Chapter	Content	Rough Size
1	introduction/concepts	10 pages
2	describe what the application does	10 pages
3	describe special files or features a first-time user needs to know about	5 pages
4	provide a brief summary of frequently used commands	20 pages
5	describe how to perform a task for which the application is intended	20 pages
6	provide a working example with commands the reader can type to obtain results	10 pages
7	describe other typical tasks and give working examples	10 pages

Table 3.5
Sample Hardware User Guide Outline

Chapter	Content	Rough Size
1	introduction	1 page
2	commonly used terms	1 page
3	controls and indicators	8 pages
4	setup mode	16 pages
5	editing	4 pages
6	self-testing the xyz terminal	2 pages
7	what to do in the event of a problem	4 pages
8	installation, interface information, and specifications	9 pages
9	programmer information	37 pages
10	xyz terminal options	2 pages
Appendices:		
A	ANSI definitions and notation, glossary	3 pages
B	ASCII code chart	1 page
C	fill character requirements	1 page

Table 3.6
Sample Utility Manual Outline

Chapter	Content	Rough Size
	preface/new and changed features	2 pages
1	sort/merge description:	2 pages
	sorting records	5 pages
2	merging files	5 pages
	collating sequence	5 pages
	sort/merge specification file	4 pages
	optimization	8 pages
	summary of commands	4 pages
2	sort/merge usage summary	10 pages
3	sort/merge qualifiers:	1 page
	sequence qualifiers	4 pages
	input file qualifiers	4 pages
	output file qualifiers	5 pages
	specification file qualifiers	5 pages
	index	6 pages

Table 3.7
Sample Database User Guide Outline

Chapter	Content	Rough Size
1	an overview of the product	10 pages
2	what does the product require?	2 pages
	application files	5 pages
	sources of data	5 pages
	runtime system	5 pages
3	how to create a database application:	2 pages
	databases	4 pages
	connections to data sources	4 pages
	forms and reports	5 pages
	menus	2 pages
	help and error messages	2 pages
4	features of database applications:	6 pages
	menus	
	forms/reports	
	user assistance messages	
	windows and tasks	
	security features	
	database commands	
	macros	
5	working with a database application:	20 pages
	invoking the database systems	5 pages
	choosing a set of key definitions	2 pages
	running a database application	2 pages
6	using building tools to define an application:	
	building an application file	5 pages
	building the database	10 pages
	building a form/report	2 pages
	testing your application	4 pages
	creating a help message	2 pages
	creating a macro	10 pages
	making future changes	2 pages
7	keypad designs	2 pages
	glossary	10 pages
	index	5 pages

For a database product, your user guide might contain different headings, as shown in Table 3.7.

Some hardware and software products may need programmer manuals. Your computer terminal or system may have a programmer manual that lets

you define how your screen displays information or accepts data from a computer, telephone, or modem. A software product may have a programmer manual that describes how to change the software or add new features to your package. The organization for these documents will be similar in intent to other user manuals: describe the product briefly, then provide enough information so the user can perform typical user tasks.

Cards, Wall Charts, and Posters

TIP ***Create reference cards.***

You can write *reference cards* (which are very popular) to summarize information—for example, for commands or setup information, or for language documentation to provide a quick reference to the language syntax.

TIP ***Create wall charts or posters.***

For complex syntax, you may decide to provide wall charts or posters that summarize the syntax at a glance. Syntax posters can be very useful for languages with many statements or for database systems. For a reference card, you can usually place commands or statements in alphabetical order, but sometimes your readers demand organization by task.

A novice benefits from summaries, repetition, illustrations, examples, and succinct definitions of terms, but not from the details that only the more experienced reader will want. Organize reference material for skimming or quick reference with bold heads, horizontal lines, bulleted lists, notes, and tables. A reference manual typically contains introductory information about the product and then goes immediately to the detailed reference information, which is alphabetically arranged.

Concept and Introductory Documents

For a *concept* document, treat your material as though you are writing a primer, leading from the simple to the complex.

TIP ***In a concept document, develop ideas from simple to complex.***

A concept document might have the organization shown in Table 3.8.

An *introductory* document may have more detail than a concept document. Table 3.9 shows that an introductory document for a programming language might go into quite a lot of detail.

The Quick-Start Guide

A *quick-start guide* brings your readers quickly up to speed on the features and use of a product. Intended for new users of an application, not necessarily

Table 3.8
Sample Concept Document Outline

Chapter	Content	Rough Size
	preface	3 pages
1	overview	12 pages
2	users and applications	27 pages
3	data management	24 pages
4	hardware	15 pages
5	networks	9 pages
6	services	6 pages
7	future considerations	2 pages
	glossary	7 pages
	index	5 pages

Table 3.9
Sample Introductory Language Manual Outline

Chapter	Content	Rough Size
1	introduction and basics	15 pages
2	types and variables	40 pages
3	operators and expressions	14 pages
4	control flow	15 pages
5	functions and complete programs	50 pages
6	independent compilation and data abstraction	22 pages
7	exceptions	16 pages
8	concurrent programming	17 pages
9	the C processor	10 pages
10	a database query example	7 pages
Appendices:		
A	library functions	50 pages
B	C tools	5 pages
C	ANSI standard C	1 page
D	ASCII table	1 page
E	implementation-dependent characteristics	1 page

Table 3.10
Sample Quick-Start Guide Outline

Chapter	Content	Rough Size
1	getting started	5 pages
2	starting to write	10 pages
3	creating lists and tables	15 pages
4	resolving cross-references	5 pages
5	creating a table of contents	5 pages
6	building a book	4 pages
	index	3 pages

novices, it shows how to begin to use the application. A quick-start guide typically contains working examples and leads readers through a set of exercises. Once readers have completed the exercises, they should be well along in understanding what the product does and how it can be used. A quick-start guide for a text-processing system might contain the topics shown in Table 3.10.

The Documentation Set

When dealing with a *multiple manual documentation set*, you provide cross-referencing and a master index. (A *master index* is a combined alphabetical list of all index entries from all books in the set.) The term "master index" can also mean an index like that in a large encyclopedia, such as the *Encyclopedia Britannica*, that provides extensive cross-referencing for article topics and topics within articles. Such an index is rare in computer documentation, however, it will become more common with increasing use of online methods.

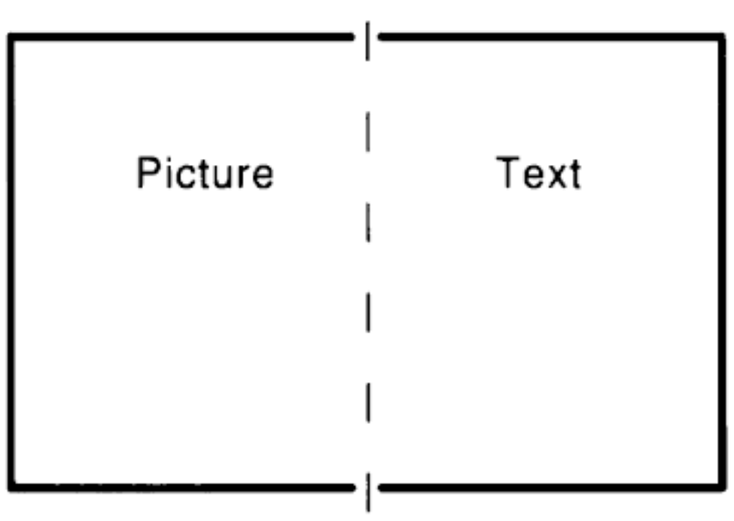
A master index is particularly valuable in a large documentation set, because readers use it to select the book in the set that contains what they seek. In general, software documentation is used much more for reference than for instruction.

The menu-driven system, for example, is very difficult to document using traditional chapter organization because a person using a menu-driven system can navigate through the system by many routes. It is impossible to represent all these routes with a traditional organization. The solution is to document the material in chunks and to ensure that the software provides *context-sensitive help*. With context-sensitive help, when the user encounters a problem and receives an error message on the screen, the user can press a special key, perhaps the PF2 or HELP key, and the software responds with appropriate instructions to correct the problem. With a software system that provides good explanatory messages in response to a user request for help, you won't need

a manual to explain recovery procedures; the procedures become part of the software.

This is an area where writers can contribute succinct messages and recovery procedures. If the software system does not offer context-sensitive help, you will have to provide your reader with lots of documentation: explicit procedures, maps of menus, and perhaps mapping of all tasks into tables with quick look-up facilities.

A special style of presenting technical information is *structured documentation*, in which you prepare information displays across the two open pages of your book. One page contains a diagram, the other contains your supporting text. This method has been very successful in hardware maintenance guides, where you show the physical component in your picture and provide the maintenance procedure in your text. The following illustration shows the page layout you use for structured documentation.



Another type of documentation is *wordless* documentation, a style of developing information that relies on pictures rather than words. Pictures can be used to illustrate procedures—for example, how to install a piece of hardware—but they have serious limitations for detailed instructions and reference information. However, as more graphics capabilities become available, you are sure to find ways to exploit this area further.

Whatever system you use for the overall organization of the information you write, you must also know and follow the usual organization of sentences and paragraphs. Sometimes online systems impose special requirements, for example, requiring you to reorganize a paragraph or section so that information critical to the readers' understanding appears early in the text. Screen displays are much shorter than printed pages, and their use can force changes of paragraph structure to accommodate your readers' needs.

Organizational Elements of Technical Documents

Technical documents must virtually always contain certain organizational elements:

- Title

- Copyright and trademarks
- Table of contents
- Chapters
- Index

On rare occasions—for example, when you prepare a reference card or a syntax poster—you don't need an index or a table of contents.

Optional elements include:

- Abstract
- Glossary
- Lists of illustrations, figures, tables, examples
- Appendices
- Bibliography
- Footnotes
- Notes (explanatory or cautionary)

Your *title* describes in a brief way the content of your document, and since it is a name, it should convey a complete thought. A title is usually a noun phrase, rarely a sentence. Create a title that reflects the subject of your document accurately. For example, the title *Write Right!* is a sentence, but *The Art of Technical Documentation* is not.

The *copyright* specifies who holds the rights to copy the document. When you work for a company, large or small, the copyright of your documents is owned by that company. If you work as a contractor for hire, the company that hires you typically owns the copyright. If you work in a freelance capacity, you may be able to negotiate for some of the copyrights to your work. Books in the public domain, for example, publications of the U.S. Government, are not copyrighted because they were paid for with taxpayer money.

The page that contains the copyright statement typically contains the names of any *trademarks* you have mentioned in your document. To protect your company against litigation by others, and to protect your company's trademarks, you must state not only what names are trademarked but also which company owns that trademark and whether or not the trademark is registered. For an example of trademark use, see the copyright page of this book.

The *table of contents* shows all the major organizational elements of your document. If you write a book divided into chapters and sections and subsections, your table of contents shows the headings you have supplied for each of these elements.

The *chapters* or paragraphs of your document form the bulk of its informational content.

An *index* provides a way for your readers to access specific topics without