Making TEX Work

Norman Walsh

O'Reilly & Associates, Inc.
90 Sherman Street Cambridge, MA 02140
`norm@ora.com`

20 April 1994

Abstract

This article contains the Table of Contents, Preface, and Chapter 4 of *Making TEX Work* by Norman Walsh; published by O'Reilly and Associates (ISBN 1-56592-051-1).
No effort has been made to preserve the 'look and feel' of the book in this text, and some simplifications have been performed (particularly with respect to special formatting issues and package names).

Title	:	Making TEX Work
Author	:	Norman Walsh
Publisher	:	O'Reilly and Associates
Date	:	April, 1994
ISBN	:	1-56592-051-1
Pages	:	518
Price	:	US $29.95

— — * — —

1 Contents 'Making TEX Work'

1.1 Table of Contents

— — ∗ — —

1.2 List of Figures

— — ∗ — —

1.3 List of Tables

— — ∗ — —

1.4 List of Examples

— — ∗ — —

2 Preface 'Making TEX Work'

TEX is a tool for creating professional quality, typeset pages of any kind. It is particularly good, perhaps unsurpassed, at typesetting mathematics; as a result, it is widely used in scientific writing. Some of its other features, like its ability to handle multiple languages in the same document and the fact that the content of a document (chapters, sections, equations, tables, figures, etc.) can be separated from its form (typeface, size, spacing, indentation, etc.) are making TEX more common outside of scientific and academic circles.

Designed by Donald Knuth in the late 1970s, more than a decade of refinement has gone into the program called 'TEX' today. The resulting system produces publication-quality output while maintaining portability across an extremely wide range of platforms.

Remarkably, TEX is free. This fact, probably as much as any other, has contributed to the development of a complete 'TEX system' by literally thousands of volunteers. TEX, the program, forms the core of this environment and is now supported by hundreds of tools.

2.1 Why Read This Book?

This book is for anyone who uses TEX. Novices will need at least one other reference, because this book does not describe the nuts and bolts of writing documents with TEX in any great detail.

If you are new to TEX, there is much to learn. There are many books that describe how to use TEX. However, the focus of this book is mostly at a higher level. After digesting Chapter 1, *The Big Picture*, you should be able to proceed through the rest of the book without much difficulty even if you have never seen TEX before. So, if you are a system administrator interested in learning enough about these programs to install and test them for your users, you should be all set. If you are interested in learning how to write documents with TEX, this book will be helpful, but it will not be wholly sufficient.

Why do you need this book at all? Although many individual components of the TEX system are well documented, there has never before been a complete reference to the whole system. This book surveys the entire world of TEX software and helps you see how the various pieces are related.

A functioning TEX system is really a large collection of programs that interact in subtle ways to produce a document that, when formatted by TEX, prints the output you want. All the different interactions that take place ultimately result in less work for you, the writer, even though it may seem like more work at first. Heck, it may *be* more work at first, but in the long run, the savings are tremendous.

Many books about TEX refer the reader to a 'local guide' for more information about previewing and printing documents and what facilities exist for incorporating special material into documents (like special fonts and pictures and figures). In reality, very few local guides exist.

The TEX environment is now mature and stable enough to support a more 'global guide.' That is what this book attempts to be. It goes into detail about previewing and printing, about incorporating other fonts, about adding pictures and figures to your documents, and about many other things overlooked by other books.

Because fonts play a ubiquitous role in typesetting, this book is also about METAFONT, the tool that Donald Knuth designed for creating fonts.

2.2 Scope of This Book

Here's how the book is laid out.

Part I: An Introduction to TEX.

Chapter 1, *The Big Picture*. If you don't know anything about TEX at all, this chapter will help you get started. If you're a system adminstrator charged with the task of installing and maintaining TEX tools, you'll get enough information to do the job.

Chapter 2, *Editing*. An overview of some environments that can make working with TEX documents easier. It describes some editors that 'understand' TEX, how to integrate TEX into a 'programmer's editor,' spellchecking, revision control, and other aspects of TEXnical editing.

Chapter 3, *Running TEX*. The mechanics of running TEX, the program. It discusses what things TEX needs to be able to run, how to start TEX, command-line options, leaving TEX, and recovery from errors.

Chapter 4, *Macro Packages*. Overview of TEX macro packages. This chapter describes how to make a format file, the major general-purpose writing packages, some special-purpose writing packages, how to make slides for presentations, and how to handle color in TEX.

Part II: Elements of a Complex Document.

Chapter 5, *Fonts*. This chapter explores the issues that need to be addressed when using fonts. Many of these issues are not particularly TEX-related, but TEX is very flexible, and it's important to understand the tradeoffs that must be made. This chapter also examines some TEX-specific issues: font selection, files that TEX needs, automatic font generation, and virtual fonts.

Chapter 6, *Pictures and Figures*. How many ways are there to include pictures and figures in a TEX document? Lots and lots of ways. This chapter examines them all.

Chapter 7, *International Considerations*. TEX is well qualified to do international typesetting. This chapter looks at the issues that are involved: representing international symbols in your input file, what TEX produces, getting the right fonts, multiple languages in the same document, and macro packages and style files that solve some of these problems. Some strategies for dealing with very difficult languages (like Japanese and Arabic) are also explored.

Chapter 8, *Printing*. What goes in has to come out. This chapter tells the what, where, why, and how of printing your documents.

Chapter 9, *Previewing*. Save paper; preview your documents before you print them.

Chapter 10, *Online Documentation*. Online documentation is becoming increasingly popular. This chapter explores different ways that both typeset and online documentation can be produced from the same set of input files.

Chapter 11, *Introducint METAFONT*. Sometimes it is necessary or desirable to create a special version of a standard TEX font. Maybe you really need the standard 10pt font at 11.3pt. This chapter will tell you how to work with existing METAFONT fonts. It *won't* tell you how to create your own fonts; that's a whole different story.

Chapter 12, *Bibliographies, Indexes, and Glossaries*. Maintaining a bibliographic database can be a great timesaver. This chapter looks at the BibTEX program and other tools for building and using bibliographic databases. It also discusses the creation of indexes and glossaries.

Part III: A Tools Overview.

Chapter 13, *Non-commercial Environments*. Many TEX environments are freely available. This chapter describes public domain, free, and shareware versions of TEX.

Chapter 14, *Commercial Environments*. A large, complex system like TEX can be overwhelming (although I hope less so after you read this book ;-). One of the advantages of selecting a commercial implementation of TEX is that some form of customer support is usually provided. Still other commercial implementations offer features not found in any free releases. This chapter describes several commercial TEX releases.

Chapter 15, *TEX on the Macintosh*. Many issues discussed in this book apply equally to all platforms, including the Macintosh platform, but the Mac has its own special set of features. This chapter looks at some versions of TEX and other tools designed specifically for use on the Mac.

Chapter 16, *TEX Utilities*. This chapter lists many of the the utilities available in the CTAN archives and provides a brief description of what they do.

Appendix A, *Filename Extension Summary*. Lots of files can be identified by their extensions. This appendix lists the extensions that are most often seen in conjunction with TEX and describes what the associated files contain.

Appendix B, *Font Samples*. Examples of many METAFONT fonts available from the CTAN archives.

Appendix C, *Resources*. A complete list of the resources described in this book.

Appendix D, *Long Examples*. This appendix contains examples (scripts, batch files, programs) that seemed too long to place in the running text.

Bibliography. Where I learned what I know. Also, where you can look for more information about specific topics.

2.3 Conventions Used in This Book

The following typographic conventions are used in this book:

Italic is used for filenames, directories, user commands, program names, and macro packages (except where the distinctive logo type is used). Sometimes italics are used in the traditional way for emphasis of new ideas and important concepts.

`Typewriter` is used for program examples, FTP sites, TEX control sequences, and little bits of TEX syntax that appear in running text (for example, typewriter text in a reference to the LATEX `picture` environment is a clue that LATEX literally uses the word 'picture' to identify this environment).

`Typewriter Bold` is used in examples to show the user's actual input at the terminal.

`Typewriter Italic` identifies text that requires a context-specific substitution. For example, *`filename`* in an example would be replaced by some particular filename.

Footnotes are used for parenthetical remarks. Sometimes, lies are spoken to simplify the discussion, and the footnotes restore the lie to the truth.[1]

Filename extensions, like '*.tex*' in *book.tex*, are shown in uppercase letters when referring to a particular type of file. For example, a TEX Font Metric or TFM file would be *somefile.tfm*. The actual extension of the file may be different (upper or lowercase, longer or shorter) depending on your operating environment.

[1] And sometimes they don't ;-)

When the shell prompt is shown in an example, it is shown as a dollar sign, `$`. You should imagine that this is your system’s prompt, which might otherwise be `%`, `C>`, `[C:\]`, or a dialog box.

When spaces are important in an example, the ‘ ’ character is used to emphasize the spaces. Effective, isn’t it?

In some places, I refer to specific keys that you should press. When it’s important that I mean pressing particular keys and not typing something, I emphasize the keys. For instance, an example that includes Enter means that you should literally press the Enter or Return key. The sequence Ctrl-D means that you should press and hold the ‘Control’ and ‘d’ keys simultaneously. Control-key combinations aren’t case sensitive, so you needn’t press the shift key.

2.4 How to Get TEX

TEX and the other programs mentioned in this book are available from a number of places. It’s impossible to list all of the places where you might find any given tool, but there is one place where you will almost certainly find *every* tool: the Comprehensive TEX Archive Network (CTAN).

This network is a fully-mirrored anonymous FTP hierarchy on three continents. Always use the FTP site that is geographically closest to you. The following table lists the current members of CTAN as of July, 1993:

Geographic Location	Site	IP Address	Top Level Directory
United States	`ftp.shsu.edu`	192.92.115.10	*/tex-archive*
England	`ftp.tex.ac.uk`	131.151.79.32	*/tex-archive*
Germany	`ftp.uni-stuttgart.de`	129.69.8.13	*/tex-archive*

You may also access the CTAN archives by electronic mail if you do not have FTP access. For up-to-date instructions about the mail server, send the single-line message `help` to *fileserv@shsu.edu*.

2.4.1 Where Are the Files?

Every CTAN mirror site has the same well-organized directory structure. The top-level directory also contains a complete catalog of current files organized by name, date, and size. The catalogs are named *FILES.byname*, *FILES.bydate*, and *FILES.bysize*, respectively, in the top level directory. The top-level directory contains the following subdirectories:

The archives at `ftp.shsu.edu` and `ftp.tex.ac.uk` also support *gopher* access to the archives. The UK *gopher* supports indexed access to the archives. A World Wide Web (hypertext) interface to the archives is available from:

```
http://jasper.ora.com/CTAN/ctan.html
```

This interface includes brief descriptions of many packages and the ability to perform keyword and date searches.

Directory	Description of Contents
tools	Archiving tools (*unzip*, *tar*, *compress*, etc.)
biblio	Tools for maintaining bibliographic databases
digests	Electronic digests (TEXhax, UKTEX, etc.)
info	Free documentation, many good guides
dviware	Printing and previewing software
fonts	Fonts for TEX
graphics	Software for working with pictures and figures
help	Online help files, etc.
indexing	Indexing and glossary building tools
language	Multi-national language support
macros	Macro packages and style files
misc	Stuff that doesn’t fit in any other category
support	Tools for running and supporting TEX
systems	OS-specific programs and files
web	Sources for TEX programs (in Web)

2.4.2 Getting Software Without FTP

The electronic alternatives to FTP, described in the section ‘Getting Examples From This Book’ of this chapter are also viable alternatives for getting software from the CTAN archives.

In addition, there are a number of ways to get distributions through nonelectronic channels. The names and addresses of these sources are listed in Appendix C, *Resources*.

You can get many of the popular TEX distributions on diskette from the TEX Users Group (TUG). Emacs, *Ghostscript*, and other packages by the Free Software Foundation (FSF) are available on tape directly from the FSF. You may also find large bulletin board systems that support TEX (for example, Channel1 in Cambridge, MA)

2.4.3 Getting Examples From This Book

All of the substantial code fragments and programs printed in this book are available online. The examples in this book are all in *Perl*, a language for easily manipulating text, files, and processes. I decided to use *Perl* simply because it is available for every platform discussed in this book. It is the only ‘universal’ scripting language that will work under MS-DOS, OS/2, UNIX, and the Macintosh. All of the scripts in this book can be converted to a different scripting language (the various UNIX shells or something like *4DOS*’s extended batch language for MS-DOS and OS/2) if you prefer. I’ve tried to write the *Perl* scripts in a straightforward way so that any given task won’t be too difficult.

The examples are available electronically in a number of ways: by FTP, FTPMAIL, BITFTP, and UUCP. The cheapest, fastest, and easiest ways are listed first. If you read from the top down, the first one that works is probably the best. Use FTP if you are directly on the Internet. Use FTPMAIL if you are not on the Internet but can send and receive electronic mail to Internet site (this includes CompuServe users). Use BITFTP if you send electronic mail via BITNET. Use UUCP if none of the above work.

NOTE: The examples were prepared using a UNIX system. If you are running UNIX, you can use them without modification. If you are running on another platform, you may need to modify these examples to correct the end-of-line markers. For example, whereas under UNIX every line ends with a line feed character (the carriage return is implicit), under DOS every line must end with explicit carriage return and line feed characters.

FTP To use FTP, you need a machine with direct access to the Internet. A sample session is shown below.

```
$ ftp ftp.uu.net
Connected to ftp.uu.net.
220 ftp.UU.NET FTP server
   (Version 6.34 Oct 22 14:32:01 1992) ready.
Name (ftp.uu.net:prefect): anonymous
331 Guest login ok, send e-mail address as
    password.
Password: prefect@guide.com
230 Guest login ok, access restrictions apply.
ftp> cd /published/oreilly/nutshell/maketexwork
250 CWD command successful.
ftp> get README
200 PORT command successful.
150 Opening ASCII mode data connection for
    README (xxxx bytes).
226 Transfer complete.
local: README remote: README
xxxx bytes received in xxx seconds (xxx Kbytes/s)
ftp> binary
200 Type set to I.
(Repeat get commands for the other files.
  They are listed in the README file.)
ftp> quit
221 Goodbye.
$
```

FTPMAIL FTPMAIL is a mail server available to anyone who can send electronic mail to, and receive it from, Internet sites. This includes most workstations that have an email connection to the outside world and CompuServe users. You do not need to be directly on the Internet.

Send mail to *ftpmail@decwrl.dec.com*. In the message body, give the name of the anonymous FTP host and the FTP commands you want to run. The server will run anonymous FTP for you and mail the files back to you. To get a complete help file, send a message with no subject and the single word `help` in the body. The following is an example mail session that should get you the examples. This command sends you a listing of the files in the selected directory and the requested example files. The listing is useful if there's a later version of the examples you're interested in.

```
$ mail ftpmail@decwrl.dec.com
Subject: reply prefect@guide.com
connect ftp.uu.net
chdir /published/oreilly/nutshell/maketexwork
dir
get README
quit
```

A signature at the end of the message is acceptable as long as it appears after `quit`.

BITFTP BITFTP is a mail server for BITNET users. You send it electronic mail messages requesting files, and it sends you back the files by electronic mail. BITFTP currently serves only users who send it mail from nodes that are directly on BITNET, EARN, or NetNorth. BITFTP is a public service of Princeton University.

To use BITFTP, send mail containing your FTP commands to *BITFTP@PUCC*. For a complete help file, send `HELP` as the message body.

The following is the message body you should send to BITFTP:

```
FTP ftp.uu.net
NETDATA USER anonymous
PASS your Internet e-mail address (not your BIT-
NET address)
CD /published/oreilly/nutshell/maketexwork
DIR
GET README
QUIT
```

Questions about BITFTP can be directed to *MAINT@PUCC* on BITNET.

UUCP UUCP is standard on virtually all UNIX systems and is available for IBM-compatible PCs and Apple Macintoshes. The examples are available by UUCP via modem from UUNET; UUNET's connect-time charges apply. You can get the examples from UUNET whether you have an account or not. If you or your company has an account with UUNET, you will have a system with a direct UUCP connection to UUNET. Find that system, and type (as one line):

```
$ uucp uunet\!~/published/oreilly/
                  nutshell/maketexwork/README \
  yourhost\!~/yourname
```

The README file should appear some time later (up to a day or more) in the directory */usr/spool/uucppublic/yourname*. If you don't have an account, but would like one so that you can get electronic mail, contact UUNET at 703-204-8000.

If you don't have a UUNET account, you can set up a UUCP connection to UUNET in the United States using the phone number 1-900-468-7727. As of this writing, the cost is 50 cents per minute. The charges will appear on your next telephone bill. The login name is *uucp* with no password. Your entry may vary depending on your UUCP configuration.

Gopher If you are on the Internet, you can use the *gopher* facility to learn about online access to examples through the O'Reilly Online Information Resource. Access *gopher.ora.com* as appropriate from your site.

2.5 Versions of TEX

The most recent versions of TEX and METAFONT are version 3.1415 and version 2.71, respectively. Version 3 of TEX introduced several new features designed to improve support for non-English languages (including the use of

8-bit input and some refinements to hyphenation control). If you use an older version of TEX, you should upgrade.

Donald Knuth has specified that TEX's version number converges to π, therefore version 3.1415 is only the fourth minor revision after version 3. The next minor revision will be version 3.14159. Similarly, METAFONT's version number converges to e (2.7182818284...).

2.6 Implementations and Platforms

The interface that TEX presents to the writer is very consistent. Most of the examples described in this book are applicable to every single implementation of TEX. However, TEX is not a closed system. It is possible to step outside of TEX to incorporate special elements into your document or take advantage of the special features of a particular environment. These extensions can dramatically restrict the portability of your documents.

Many of the topics covered in this book offer alternatives in those areas that are less portable. Therefore, it is natural to ask what implementations are really covered.

Before outlining which implementations are covered, let me suggest that this book will be useful even if you are using an implementation not 'officially' covered here. The reality of the situation is this: many, many tools have been ported with TEX. Many of the tools mentioned in this book are available on platforms that are not specifically discussed. Time and equipment constraints prevented Amiga, Atari, NeXT, VMS, and Windows NT implementations of TEX from being specifically addressed in this edition of the book.

2.6.1 UNIX

UNIX is probably the most common TEX platform. The emphasis in this book is on UNIX workstations running X11, producing output for PostScript and HP LaserJet printers.

Linux and other personal computer implementations of UNIX are not addressed specifically; however, with the successful port of X11 to *Linux*, I'm confident that every UNIX tool here can be, or has been, ported to *Linux* (and probably other PC UNIX environments).

The only implementation of the TEX program for UNIX considered in any detail is the free implementation distributed in *web2c*. This distribution is described in the section called 'Web2C' in Chapter 13, *Non-commercial Environments*. Most of the other UNIX tools discussed here are also free.

2.6.2 MS-DOS

With very few exceptions, the tools in this book are available under MS-DOS. Because PCs are very popular, a lot of effort has gone into porting UNIX tools to MS-DOS. Some packages, however, require a 386SX (or more powerful) processor. For the most part, I focus on PCs running MS-DOS only; however, Microsoft Windows and DesqView are not entirely ignored.

There are quite a few options when it comes to selecting an implementation of the TEX program under MS-DOS. Several free implementations are discussed as well as some commercial implementations. For more information about these implementations, consult Chapter 13, *Non-commercial Environments*, and Chapter 14, *Commercial Environments*.

2.6.3 OS/2

In this book, OS/2 is treated primarily as a superset of MS-DOS. When possible, I look at OS/2-specific versions of each utility, but rely on MS-DOS as a fall-back.

Extensions to emTEX for OS/2 are explored, as are editing environments such as *epm*. The multi-threaded nature of OS/2 allows more complete porting of UNIX tools. When better ports are available for OS/2, they are discussed.

2.6.4 Macintosh

The Macintosh is very different from the systems described above. Chapter 15, *TEX on the Macintosh*, discusses the Macintosh environment in detail.

There are four implementations of TEX for the Macintosh. Three are freely available, and one is commercial: CMacTEX is free, OzTEX and DirectTEX are shareware, and *Textures* is a commercial package from Blue Sky Research.

2.7 We'd Like to Hear From You

We have tested and verified all of the information in this book to the best of our ability, but you may find that features have changed (or even that we have made mistakes!). Please let us know about any errors you find, as well as your suggestions for future editions, by writing:

O'Reilly & Associates, Inc.
103 Morris Street, Suite A
Sebastopol, CA 95472
1-800-998-9938 (in the US or Canada)
1-707-829-0515 (international/local)
1-707-829-0104 (FAX)

You can also send us messages electronically. To be put on the mailing list or request a catalog, send email to:

nuts@ora.com	(via the Internet)
uunet!ora!nuts	(via UUCP)

To ask technical questions or comment on the book, send email to:

bookquestions@ora.com (via the Internet)

2.8 Acknowledgments

This book would not exist if I had not received support and encouragement from my friends and colleagues, near and far. I owe the deepest debt of gratitude to my wife, Deborah, for patience, understanding, and support as I progressed through what is easily the most all-consuming task I have ever undertaken.

The earliest draft of this book came about because my advisor at the University of Massachusetts, Eliot Moss, allowed me to tinker with the TEX installation in the Object Systems Lab and was always able to suggest ways to make it better. My friends and colleages at UMass, Amer Diwan, Darko Stefanović, Dave Yates, Eric Brown, Erich Nahum, Jody Daniels, Joe McCarthy, Ken Magnani, Rick Hudson, and Tony Hosking, asked all the hard questions and didn't seem to mind when I used them as guinea pigs for my latest idea.

I'm indebted also to Eberhard Mattes, Geoffrey Tobin George D. Greenwade, Peter Schmitt, Sebastian Rahtz, and Tomas Rokicki, who provided technical review comments on the materials presented here. Jim Breen and Ken Lunde offered invaluable feedback on Chapter 7.

And I'd like to thank a lot of people at O'Reilly for their help and enthusiasm; in particular, my editor, Debby Russell, offered advice, helpful criticism, and support beyond the call of duty (Debby keyed most of the index for this book as production deadlines drew near and other arrangements fell through); Chris Tong organized the raw entries into a usable index; Lenny Muellner, Donna Woonteiler, and Sheryl Avruch allowed me to work on the book when it wasn't technically my job; Stephen Spainhour copyedited it into English with the help of Leslie Chalmers and Kismet McDonough (Stephen offered helpful suggestions along the way, too); Jennifer Niederst helped me get the design right; and Chris Reilly created the figures and screen dumps. I enjoyed working with everyone at O'Reilly so much that I left UMass and joined the production department myself ;-).

Several companies provided review copies of their software while I was writing this book. I would like to thank ArborText, Blue Sky Research, Borland International, The Kinch Computer Company, LaserGo, Personal TEX, TCI Software Research, and Y&Y, for their generosity.

Finally, I'd like to thank the entire Internet TEX community. Countless thousands of questions and answers on the Net refined my understanding of how TEX works and what it can do.

— — ∗ — —

3 Chapter 4 from book: Macro Packages

Everyone who uses TEX uses a macro package (also called a 'format'). A macro package extends TEX to provide functionality that is suited to a particular task or set of tasks.

This chapter provides a summary of TEX macro packages. General-purpose packages designed to typeset a wide range of documents—articles, books, letters, and reports—are examined first. The general-purpose packages described are Plain TEX, Extended Plain TEX, LATEX2ε, LATEX, AMSTEX, AMSLATEX, Lollipop, and TEXinfo. After surveying the general-purpose packages, several special-purpose packages designed to handle specific tasks—typesetting transparencies, music, chemistry or physics diagrams, etc.—are described. The special-purpose packages surveyed are SliTEX, FoilTEX, Seminar, MusicTEX, ChemStruct, and ChemTEX.

There are a lot of overlapping features and similar commands in the general-purpose packages. To understand why this is the case, consider how a new macro package comes into existence. An ambitious person, who is very familiar with TEX, decides that there are some things she would like to express in her documents that are difficult to express with existing formats. Perhaps, for example, no existing format produces documents that match the precise specifications required for publishing in her field, or perhaps she has in mind a whole new document structuring paradigm. A more mundane possibility is simply that she has been customizing an existing format for some time and now feels it has enough unique features to be useful to others.

In any event, a new format is born. Now, if this format is designed for a very specific task, writing multiple-choice mathematics exams, for example, it might not have very many general-purpose writing features. On the other hand, if it is designed for writing longer, more general documents (e.g., history textbooks or papers to appear in a particular journal) then there are a number of features that it is likely to include; provisions for numbered lists, cross references, tables of contents, indexes, and quotations are all examples of features common to many documents.

To support these common features, many macro packages have similar control sequences. This stems from the fact that they are all built on top of a common set of primitives and that macro package authors tend to copy some features of other packages into their own.

You may find that you'd like to use the features of several different packages in the same document. Unfortunately, there is no provision for using multiple formats to process a single document. The features required to process most documents are shared by all of the general-purpose formats, however. You are more likely to need multiple macro packages if you want to use a special format to construct a diagram or figure and incorporate it into a document. Chapter 6, *Pictures and Figures*, describes several ways to take 'electronic snippets' of one document and insert them into another, which is one possible solution to this problem.

If you're beginning to feel a little lost, have no fear. Most general-purpose formats are sufficient for most documents. And there's no reason why every document you write has to be done with the same format. Many people find LATEX and Plain TEX sufficient, but if you're writing an article for a particular journal and someone has written a format specifically for that journal's documents, by all means use it. It is more likely, however, that someone has written a style file which tailors LATEX to the requirements of the journal.

In addition to describing some common macro packages in this chapter, I'll describe how to build format files for them. If the packages that you want to use have already been installed at your site, you can ignore the installation sections.

The packages that you find most convenient will depend on the tasks you perform and how well each package suits your work style. The list of packages in this chapter is not meant to be all-inclusive, nor is it my intention to suggest which packages are best. Use the ones *you* like, for whatever reasons.

I can hear some of you already, 'I don't really need a macro package,' you say, 'I can roll my own with just TEX.'

And you are absolutely correct.

I don't recommend it, however. It's akin to using your compiler without any of the built-in functions. Most TEX primitives offer little support by themselves for writing documents.

A New Perspective: LATEX2$_\varepsilon$ (and its successor, which will be LATEX3) seek to address many of the problems mentioned above by defining a core LATEX format with extension packages to provide custom features.

The LATEX2$_\varepsilon$ system provides a single format file that supports LATEX, AMSTEX, and SliTEX. For the time being, LATEX2$_\varepsilon$ is described separately in this chapter along with the other formats. Be aware, however, that LATEX2$_\varepsilon$ is now the standard LATEX[2]and you should migrate to it as soon as possible.

3.1 Installation: Making Format Files

A format file, as described in Chapter 3, *Running TEX*, is a special 'compiled' version of a macro package. The iniTEX program interprets all of the control sequences in a macro package and writes the corresponding memory image into a file. Loading a format file is much faster than loading individual macro packages in your document because TEX does not have to interpret any of the control sequences while it is also processing text.

In general, all format files should be stored in the same directory.[4] If you install TEX in a directory called *tex*, then formats typically go in a directory called *formats* in the *tex* directory. This is not universally the case because you need separate format directories for big and small TEXs.[5]

Usually, an environment variable indicates where the format files are located. Environment variables are a common way of customizing your interaction with programs. They are usually set in your *AUTOEXEC.BAT* file for MS-DOS, *CONFIG.SYS* for OS/2, or the *rc*-file for a shell (i.e., *.cshrc*, *.kshrc*) on UNIX systems. Any good reference book for your operating system or shell will describe how to use environment variables.

A common name for the environment variable that indicates where TEX formats are located is `TEXFORMATS`. Implementations that provide big and small TEXs need another variable to indicate the directory that contains formats for big TEX.

IniTEX is not always a separate program. Some implementations of TEX combine the functionality of TEX and iniTEX into a single program and use a special switch at runtime to determine which function to perform. In this chapter, all of the examples use the program name *initex* to identify iniTEX. If you use an implementation that doesn't provide a separate iniTEX program, you should use the TEX program with the iniTEX switch instead. For example, for emTEX, use *tex /i* instead of *initex* when you build a format.

Like TEX, iniTEX needs to be able to find input files. Usually, this is accomplished by searching the directories listed in the `TEXINPUTS` environment variable. Place the input files that iniTEX needs in a directory on the `TEXINPUTS` path before running iniTEX unless otherwise directed. The `TEXINPUTS` environment variable is discussed in the 'User Files' section of Chapter 3.

Hyphenation Patterns

In order for TEX to correctly hyphenate words, every format file must contain a set of hyphenation patterns. The patterns are part of an algorithmic solution to the problem of breaking a word into syllables for hyphenation.

The details of the hyphenation algorithm (given in Appendix H of *The TEXbook*) are too complex to describe here, but two aspects of this solution deserve particular emphasis. First, using patterns means that a dictionary of hyphenated words is not necessary.[6] This saves a lot of space and time. Second, by loading different sets of patterns, TEX can achieve equal success at hyphenating any language—even English ;-). There are actually at least two sets of hyphenation patterns for English, one for British English and one for American English. Chapter 7, *International Considerations*, describes how to load multiple sets of hyphenation patterns for typesetting multilingual documents.

3.2 General-purpose Formats

This section describes several macro packages that are designed for formatting standard documents like articles or books. In order to provide some form of comparative measure, each macro package is used to create the same document, a one-page report that looks like Figure 1. I constructed this example to demonstrate a few common elements in a document: several sizes of headings, a paragraph of text, inline and displayed mathematics, and a few

[2]At the time of this writing, it's actually still in test-release, but it may be available as a standard release by the time you read this.

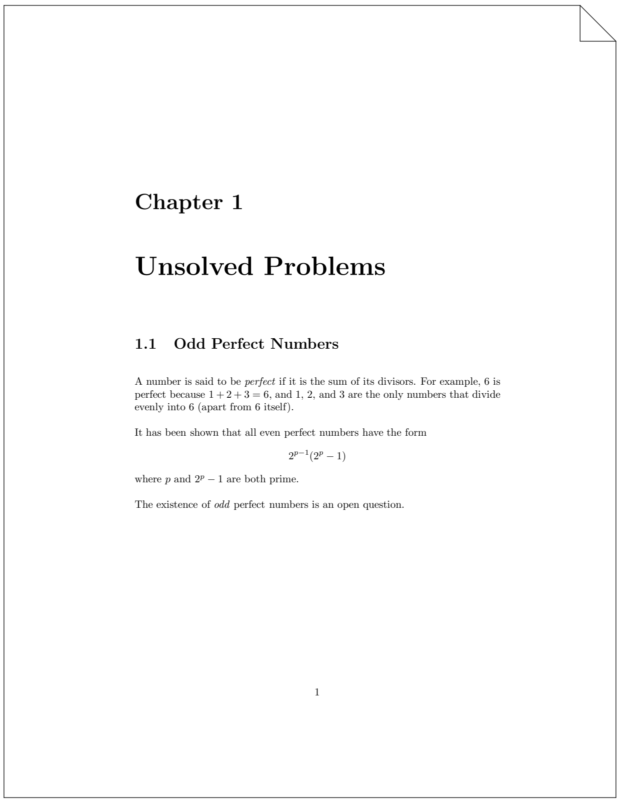
[4]On Macintosh systems and other environments that don't have directories, format files are typically stored in their own folders (or other metaphorically appropriate place ;-).

[5]The distinction between big and small TEX is described in the section called 'What Do You Run?' in Chapter 3.

[6]A small set of exceptions is maintained because the algorithm isn't perfect.

fonts. There are lots of other things that aren't shown (tables, figures, footnotes, etc.), and these elements vary as much as any other in the different macro packages.

Observant readers will notice that the examples are shown in the Computer Modern fonts while the rest of this book uses different fonts.[7] There are a number of reasons why the example is shown in Computer Modern. For one thing, all of the formats discussed here use the Computer Modern fonts by default. Using a different set of fonts would only add more complexity to each example. A more subtle problem is that I do not have appropriate mathematics fonts for Garamond. There are a number of complex issues involving the use of fonts in TEX. They are discussed in Chapter 5, *Fonts*.



Chapter 1

Unsolved Problems

1.1 Odd Perfect Numbers

A number is said to be *perfect* if it is the sum of its divisors. For example, 6 is perfect because $1+2+3=6$, and 1, 2, and 3 are the only numbers that divide evenly into 6 (apart from 6 itself).

It has been shown that all even perfect numbers have the form

$$2^{p-1}(2^p-1)$$

where p and 2^p-1 are both prime.

The existence of *odd* perfect numbers is an open question.

1

Figure 1: *Sample page*

3.2.1 Plain TEX

Plain TEX is the format written by Donald Knuth during the development of TEX. It is described fully in *The TEXbook.*

Plain TEX ties together the TEX primitives in a way that makes it practical to work in TEX. If you do not have a computer programming background, you may find Plain TEX a little bit intimidating. It is definitely a ‘roll your own’ environment. Although it demonstrably contains all of the functionality required to write everything from letters to books, there is very little ‘user-friendly’ packaging around the internals of TEX.

Aside from user interface considerations, which are highly subjective, Plain TEX lacks some functionality when compared to other formats. There is no provision in Plain TEX for automatically numbered sections, labelled figures, tables of contents, indexes, or bibliographies. Any of these functions can be constructed in Plain TEX if you are willing to invest the time and energy required to write your own macros, but they are not built into Plain TEX.

If you enjoy writing your own macros or plan to produce novel types of documents, a firm grasp of Plain TEX will allow you to write anything in TEX. A firm grasp of Plain TEX also makes it easier to understand and modify other formats (like LATEX) that are built on top of Plain TEX.

In addition, Plain TEX is the only format that is always distributed with TEX. The other formats discussed in this chapter are freely available but do not come with TEX.

The Plain TEX input that produces the report in Figure 1 is shown in Example 1.

Example 1: Plain TEX Input

```
% Format: Plain
\font\chapfont=cmbx12 scaled 1728
\font\titlefont=cmbx12 scaled 2073
\font\secfont=cmbx12 scaled 1200

\parskip=\baselineskip
\parindent=0pt
\hsize=5in
\hoffset=.75in

\leftline{\chapfont Chapter 1}
\vskip36pt

\leftline{\titlefont Unsolved Problems}
\vskip36pt

\leftline{\secfont 1.1\ \ Odd Perfect Numbers}
\vskip12pt

A number is said to be {\it perfect\/} if it
is the sum of its divisors.  For example, $6$
is perfect because $1+2+3 = 6$, and $1$, $2$,
and $3$ are the only numbers that divide
evenly into $6$ (apart from $6$ itself).

It has been shown that all even perfect numbers
have the form $$2^{p-1}(2^{p}-1)$$ where $p$
and $2^{p}-1$ are both prime.

The existence of {\it odd\/} perfect numbers is
an open question.
\bye
```

Building the Plain TEX format To build Plain TEX, you need only two files: *plain.tex* and *hyphen.tex.* These files are distributed with TEX so they should be available as soon as you have installed TEX. The *hyphen.tex* file is language-dependent. Readers who frequently work with non-English text should read Chapter 7 for more information about obtaining non-English hyphenation patterns.

The command:

```
$ initex plain \dump
```

[7] Really observant readers may have noticed that it's a version of Garamond ;-)

will create the Plain TEX format. Move the resulting files, *plain.fmt* and *plain.log*, into your TEX formats directory.

3.2.2 Extended Plain TEX

Extended Plain TEX extends Plain TEX in a number of useful ways without forcing you to use any particular 'style' of output. The argument is this: although Plain TEX really doesn't provide all of the features that you need (tables of contents, cross references, citations, enumerated lists, convenient access to verbatim input, etc), many of these features don't have any direct impact on the appearance of your document. Unfortunately, other general-purpose macro packages like LATEX, which do provide these features, tend to force you to accept their notion of what the typeset page should look like.[8]

Extended Plain TEX is an attempt to solve that problem. It provides many behind-the-scenes features without providing any general page layout commands (like `\chapter` or `\section`), which means that these features can be used inside Plain TEX without much difficulty and without changing the layout of typeset pages.

Building the Extended Plain TEX format To build the Extended Plain TEX format, you need the *plain.tex* and *hyphen.tex* files required to build the Plain format as well as the *eplain.tex* file distributed with Extended Plain TEX.[9] Make and install the Plain TEX format first, then change to the directory that contains the Extended Plain TEX distribution.

The command:

```
$ initex &plain eplain \dump
```

will create the Extended Plain TEX format. Move the resulting files, *eplain.fmt* and *eplain.log*, into your TEX formats directory.

3.2.3 LATEX2$_\varepsilon$ Versus LATEX

The tremendous popularity of LATEX in the TEX community has had an unfortunate side effect: because it is a very familiar and flexible format, many people have used it as the basis for extensions of one sort or another. This has resulted in a wide range of (slightly) incompatible formats and a lot of frustration.

This situation is being rectified by a new release of LATEX, currently called LATEX2$_\varepsilon$. The new release replaces the existing dialects of LATEX (LATEX with and without NFSS, SliTEX, AMSLATEX, etc.) with a single core system and a set of extension packages. LATEX2$_\varepsilon$ includes a compatibility mode which will allow it to continue to format existing documents without change (provided that they do not rely on local modifications to the LATEX format, of course). Local modifications can also be incorporated into the LATEX2$_\varepsilon$ system as extension packages, making LATEX2$_\varepsilon$ a complete replacement for all existing versions of LATEX and packages closely derived from LATEX.

The most significant and least compatible difference between LATEX and LATEX2$_\varepsilon$ is the font selection scheme. There are many control sequences for selecting fonts in LATEX. Some control the typeface (`\rm`, `\tt`, `\sc`, etc.); some the size (`\small`, `\normalsize`, `\large`, etc.); and some the appearance (`\it`, `\bf`, `\em`, etc.).

Under the Old Font Selection Scheme (OFSS), the control sequences for selecting a font completely override any font selection already in place. Consider, for example, the control sequences `\it` and `\bf`, which switch to italic and boldface. Using `\bf\it` produces italic text, and `\it\bf` produces boldface text, and *neither* produces boldfaced-italic text (which is probably what you wanted).

Under the New Font Selection Scheme, typeface (called *family* in NFSS parlance), appearance (called *series* and *shape*), and size are viewed as orthogonal components in font selection. Because these parameters are independent, selecting an italic appearance with the `\it` control sequence switches to italic in the current typeface and size. Under the NFSS, `\bf\it` *does* select boldface-italic in both the current typeface and size (if it is available).

LATEX2$_\varepsilon$ supports only the NFSS, version 2 (called NFSS2). For more than a year, the NFSS (initially version 1, and more recently version 2) has been available as an extension for LATEX 2.09. However, in light of stable test releases of LATEX2$_\varepsilon$, the NFSS2 package for LATEX 2.09 has been withdrawn. The discussion of NFSS in this book applies equally well to LATEX 2.09 with NFSS2, but it is described in terms of LATEX2$_\varepsilon$ in an effort to be more applicable in the future. The NFSS is discussed in more detail in Chapter 5.

3.2.4 LATEX2$_\varepsilon$

Leslie Lamport's LATEX format is probably the most commonly used TEX format. It is described in *LATEX: A Document Preparation System* and many other TEX books. LATEX2$_\varepsilon$ is the new standard LATEX. It is described in *The LATEX Companion*. The next edition of *LATEX: A Document Preparation System* will also describe LATEX2$_\varepsilon$.

At the time of this writing, LATEX2$_\varepsilon$ is available only in a test release, but by the time you read this, it is likely to be available as the new standard LATEX. It is described first in this chapter to emphasize that you should begin using LATEX2$_\varepsilon$ as soon as possible. Once LATEX2$_\varepsilon$ is out of testing, it will become LATEX, and support for older versions will not be provided (at least not by the LATEX developers). LATEX2$_\varepsilon$ includes a compatibility mode for old LATEX documents so the transition should be relatively painless.

[8]Of course, that's not strictly true. You can change the page layout of LATEX (and most other packages) to be almost anything, but it does require learning a lot about the macro package. If you are already familiar with Plain TEX (or some other Plain TEX-derived package), you probably have a set of macros that produce documents in the style you like. Why reinvent the wheel?

[9]Extended Plain TEX is available in the *macros/eplain* directory in the CTAN archives.

The central theme of LATEX is 'structured document preparation.' An ideal LATEX document is described entirely in terms of its structure: chapters, sections, paragraphs, numbered lists, bulleted items, tables, figures, and all the other elements of a document are identified descriptively. For example, you enclose figures in a figure *environment* identified by the control sequences `\begin{figure}` and `\end{figure}`.

When you are ready to print your document, select an appropriate document class, and LATEX formats your document according to the rules of the selected style. In the case of the ideal document, it might first be printed in a magazine or newsletter using the article class. Later, when it is incorporated into a book, selecting the book class is *all* that is required to produce appropriate output; the document itself is unchanged.

LATEX is written on top of Plain TEX. This means that almost any control sequence or macro that you learn about in Plain TEX can also be used in LATEX. Of course, LATEX insulates you from many Plain TEX commands by wrapping a much more user-friendly interface around them.

The LATEX2$_\varepsilon$ input that produces the sample page in Figure 1 is shown in Example 2. The only difference between this document and an old LATEX document is the use of the `\documentclass` declaration instead of the `\documentstyle` declaration.[10] For more complex documents, other minor changes may also be necessary.

Example 2: LATEX2$_\varepsilon$ Input

```
% Format: LaTeX2e
\documentclass{report}

\setlength{\parskip}{\baselineskip}
\setlength{\parindent}{0pt}

\begin{document}

\chapter{Unsolved Problems}

\section{Odd Perfect Numbers}

A number is said to be \emph{perfect} if it
is the sum of its divisors.  For example,
$6$ is perfect because \(1+2+3 = 6\), and
$1$, $2$, and $3$ are the only numbers that
divide evenly into $6$ (apart from 6 itself).

It has been shown that all even perfect numbers
have the form \[2^{p-1}(2^{p}-1)\] where $p$
and \(2^{p}-1\) are both prime.

The existence of \emph{odd} perfect numbers is
an open question.
\end{document}
```

Building the LATEX2$_\varepsilon$ format The LATEX2$_\varepsilon$ distribution is available from the directory *macros/latex2e/core* in the CTAN archives.

The following steps will build the LATEX2$_\varepsilon$ format. For more complete installation instructions, read the file *install.l2e* in the LATEX2$_\varepsilon$ distribution.

1. Place the LATEX2$_\varepsilon$ distribution in a temporary directory and make that directory the current directory. After the installation is complete, you will need to move only selected files into the standard places.
2. Restrict to only the current directory the directories that TEX searches for input files
   This can usually be accomplished by setting the environment variable `TEXINPUTS`[11] to a single period or the absolute path name of the current directory.
3. Copy the *hyphen.tex* file from the Plain TEX distribution into the current directory.
4. Issue the command:
   `$ initex unpack2e.ins`
   This will unpack all of the distribution files.
5. Build the format file by issuing the command:
   `$ initex latex2e.ltx`
   Move the resulting files *latex2e.fmt* and *latex2e.log* into the TEX formats directory.
6. In addition to the files needed to build the format, unpacking the LATEX2$_\varepsilon$ distribution creates many files that are needed for formatting documents. These files must be placed in a location where TEX will find them. However, in order to maintain a functioning LATEX 2.09 system, you must not place the new files in the same input directory as the existing files.[12]
   Create a new directory (or folder) for the new files. On the UNIX system that I use, where existing input files are stored in a directory called */usr/local/lib/tex/inputs*, I created */usr/local/lib/tex/latex2e* to store the new files. You will have to add the new directory to the *front* of the list of directories that TEX searches for input files whenever you format a document with LATEX2$_\varepsilon$.
   Move the files that the installation script produces into the new directory. Move the files *docstrip.tex*, *latexbug.tex*, *sfontdef.ltx*, *slides.ltx*, *testpage.tex*, and all of the files that end in *.cfg*, *.cls*, *.clo*, *.def*, *.fd*, and *.sty*. You should also move the files *gglo.ist* and *gind.ist* someplace where *MakeIndex* can find them. (*MakeIndex* is described in Chapter 12 *Bibliographies, Indexes, and Glossaries*.)
   One of the aspects of the test releases that continues to change is the exact list of files that must be moved. Consult the *install.l2e* file in the distribution for the exact list. The list above is from the test version of January 28, 1994.

3.2.5 LATEX

This section briefly covers LATEX version 2.09. This version of LATEX is still very widely used but it is being phased out.

[10] LATEX2$_\varepsilon$ will process documents that use `\documentstyle` in LATEX 2.09 compatibility mode.

[11] Under emTEX, this variable is called `TEXINPUT`. On the Macintosh, file searching is frequently controlled by a configuration file or dialog box.

[12] Strictly speaking, this is only true for files that have the extension *.sty* because the old version of LATEX will not attempt to use the other files.

The LATEX input to produce the sample page in Figure 1 is shown in Example 3.

Example 3: LATEX Input File

```
% Format: LaTeX
\documentstyle{report}

\setlength{\parskip}{\baselineskip}
\setlength{\parindent}{0pt}

\begin{document}

\chapter{Unsolved Problems}

\section{Odd Perfect Numbers}

A number is said to be {\em perfect\/} if it
is the sum of its divisors.  For example,
$6$ is perfect because \(1+2+3 = 6\), and $1$,
$2$, and $3$ are the only numbers that divide
evenly into $6$ (apart from 6 itself).
                %<TeX_Marker>

It has been shown that all even perfect numbers
have the form \[2^{p-1}(2^{p}-1)\] where $p$
and \(2^{p}-1\) are both prime.

The existence of {\em odd\/} perfect numbers is
an open question.
\end{document}
```

Support for the NFSS in LATEX 2.09 has been withdrawn. If you need to build a format with support for NFSS, consult the 'LATEX2$_\varepsilon$' section of this chapter.

Building the LATEX format with the OFSS The LATEX distribution[13] includes three subdirectories, *sty*, *doc*, and *general*. All of the LATEX files required to build the format file are in the *general* subdirectory. You will also need the *hyphen.tex* file required to build the Plain format.

In the *general* subdirectory, the command:

```
$ initex lplain
```

will create the LATEX format. Move the resulting files, *lplain.fmt* and *lplain.log*, to the TEX formats directory. In order to complete the installation, copy the files from the *sty* directory in the LATEX distribution into a directory where TEX searches for input files.

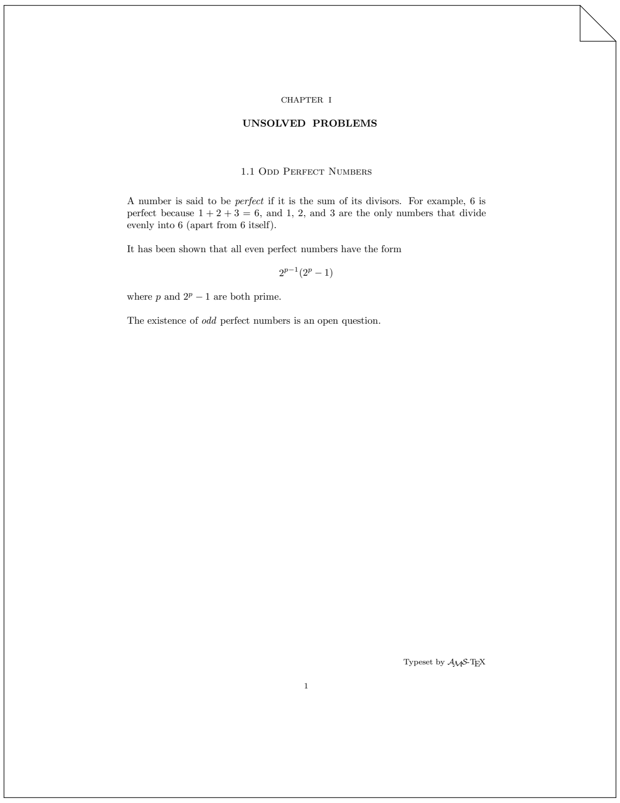
3.2.6 AMSTEX

When the American Mathematical Society selected TEX as a document preparation system, they decided to extend it in a number of ways to make the creation of papers and journals easier. They had two goals: to make it easier for authors to write mathematical papers in TEX and to make the resulting papers conform to a particular set of style specifications. AMSTEX is described completely in *The Joy of TEX*.

AMSTEX provides many commands that resemble LATEX environments. These have the form `\environment` ... `\endenvironment`. In addition, AMSTEX provides the notion of a document style to control style-related formatting issues.

Another important contribution made by the American Mathematical Society when creating AMSTEX was the construction of a large number of new fonts. The American Mathematical Society provides fonts with many more mathematical symbols than the fonts that come with TEX. These fonts are available as a separate package and can be used with any TEX macro package, not just AMSTEX.

The AMSTEX input required to produce the document in Figure 1 is shown in Example 4. The result of formatting this document does not appear exactly like Figure 1 because AMSTEX uses the style conventions of the American Mathematical Society.



CHAPTER I

UNSOLVED PROBLEMS

1.1 Odd Perfect Numbers

A number is said to be *perfect* if it is the sum of its divisors. For example, 6 is perfect because $1 + 2 + 3 = 6$, and 1, 2, and 3 are the only numbers that divide evenly into 6 (apart from 6 itself).

It has been shown that all even perfect numbers have the form

$$2^{p-1}(2^p - 1)$$

where p and $2^p - 1$ are both prime.

The existence of *odd* perfect numbers is an open question.

Typeset by $\mathcal{AMS}$-TEX

1

Figure 2: *AMS sample page*

Example 4: AMSTEX Input File

```
% Format: AMSTeX
\documentstyle{amsppt}

\parindent=0pt
\parskip=\baselineskip
\hoffset=.75in

\topmatter
\title \chapter{1} Unsolved Problems\endtitle
\endtopmatter

\document
```

[13] LATEX is available in the *macros/latex/distribs/latex* directory in the CTAN archives.

```
\head{1.1} Odd Perfect Numbers\endhead

A number is said to be {\it perfect\/} if it
is the sum of its divisors.  For example, $6$ is
perfect because $1+2+3 = 6$, and $1$, $2$, and $3$
are the only numbers that divide evenly into $6$
(apart from $6$ itself).

It has been shown that all even perfect numbers
have the form $$2^{p-1}(2^{p}-1)$$ where $p$
and $2^{p}-1$ are both prime.

The existence of {\it odd\/} perfect numbers is
an open question.
\enddocument
```

Building the AMSTEX format In order to build the AMSTEX format, you need the *plain.tex* and *hyphen.tex* files from Plain TEX and the *amstex.ini* and *amstex.tex* files from the AMSTEX distribution.

The command:

```
$ initex amstex.ini
```

will create the AMSTEX format. Move the resulting files, *amstex.fmt* and *amstex.log*, into your TEX formats directory.

3.2.7 AMSLATEX

AMSLATEX, like AMSTEX, provides many features to make typesetting mathematics convenient while meeting the standards of the American Mathematical Society for publication. However, AMSTEX lacks many of the features that are present in LATEX, like automatically numbered sections and tools for creating tables of contents and indexes.

When LATEX gained popularity, many authors requested permission to submit articles to the American Mathematical Society in LATEX. In 1987, the American Mathematical Society began a project to combine the features of AMSTEX with the features of LATEX. The result is AMSLATEX.

AMSLATEX provides all of the functionality of LATEX because it is an extension of LATEX. It also provides the functionality of AMSTEX in LATEX syntax and access to additional mathematical constructs and math symbols not present in LATEX. .

The input required to produce Figure 1 is not shown because they do not differ significantly from the LATEX sample. Because the sample document doesn't use any of AMSLATEX's additional features, it is exactly the same as the LATEX document.

Building the AMSLATEX format The AMSLATEX macros are embodied entirely in style files for LATEX. It is not necessary to build a special format file. However, the AMSLATEX macros require the New Font Selection Scheme (NFSS). Consult the section on LATEX, above, for instructions on building the LATEX format with NFSS.

3.2.8 Lollipop

It can be argued that LATEX has the following deficiency: although there are many different style options available, it is not easy for a novice user to change a style option. Changing the internals of most LATEX style options requires a deep understanding of TEX.

The Lollipop format is very different from the other formats. The central thrust of Lollipop is that it should be easy to change and customize document styles. All Lollipop documents are built from five different generic constructs: headings, lists, text blocks, page grids, and external items.

The Lollipop input required to produce a document like the sample page in Figure 1 is shown in Example 5.

Example 5: Lollipop Input File

```
% Format: Lollipop
\DefineHeading:Chapter
   breakbefore:yes whiteafter:12pt
   line:start PointSize:20 Style:bold
      literal:Chapter Spaces:1  ChapterCounter
      line:stop
   vwhite:36pt
   line:start PointSize:24 Style:bold title
      line:stop
   vwhite:24pt
   Stop
\DefineHeading:Section
   whitebefore:{20pt plus 2pt} whiteafter:14pt
   line:start PointSize:14 Style:bold
      ChapterCounter . SectionCounter
      Spaces:1 title line:stop
   label:start ChapterCounter . SectionCounter
      label:stop
   Stop
\GoverningCounter:Section=Chapter
\AlwaysIndent:no
\Distance:parskip=12pt
\Distance:hoffset=.75in
\Distance:voffset=.5in
\Start
\Chapter Unsolved Problems

\Section Odd Perfect Numbers

A number is said to be {\it perfect\/} if it
is the sum of its divisors.  For example,
$6$ is perfect because $1+2+3 = 6$, and
$1$, $2$, and $3$ are the only numbers that
divide evenly into $6$ (apart from $6$ itself).

It has been shown that all even perfect numbers
have the form $$2^{p-1}(2^{p}-1)$$ where $p$
and $2^{p}-1$ are both prime.

The existence of {\it odd\/} perfect numbers is
an open question.
\Stop
```

Building the Lollipop format To make the Lollipop format, you need the Lollipop distribution and the file *hyphen.tex* from the Plain TEX distribution.

This command will create the Lollipop format:

```
$ initex lollipop \dump
```

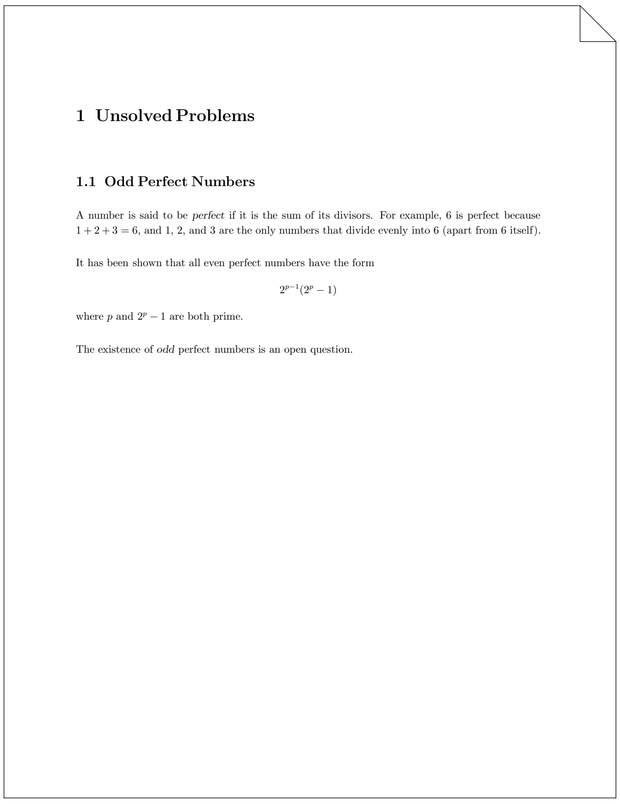
It should be performed in the directory where you installed the Lollipop distribution so that all of the Lollipop files can be located. Move the resulting files *lollipop.fmt* and *lollipop.log*, into your TEX formats directory.

3.2.9 TEXinfo

The TEXinfo format is a general-purpose format, but it was designed to support a particular application: to produce both online documentation and professional quality typeset documentation from the same source file. It is discussed in more detail in Chapter 10 *Online Documentation*.

The input file shown in Example 6 produces the typeset output shown in Figure 3. The input file for this example is complicated by the fact that it contains mathematics. None of TEX's sophisticated mechanisms for handling mathematics are applicable to plain ASCII online documentation. The online documentation produced by the example in Example 6 is shown in Figure 4.

The TEXinfo format is the official documentation format of the Free Software Foundation (FSF). Although less commonly used, a LATEX variant called LATEXinfo is also available.



1 Unsolved Problems

1.1 Odd Perfect Numbers

A number is said to be *perfect* if it is the sum of its divisors. For example, 6 is perfect because $1+2+3=6$, and 1, 2, and 3 are the only numbers that divide evenly into 6 (apart from 6 itself).

It has been shown that all even perfect numbers have the form

$$2^{p-1}(2^p-1)$$

where p and 2^p-1 are both prime.

The existence of *odd* perfect numbers is an open question.

Figure 3: *TEXinfo sample page*

Example 6: TEXinfo Input

```
\input texinfo  @c -*- TeXinfo -*-
@setfilename perf-inf.inf
@ifinfo
  @paragraphindent 0
@end ifinfo
@iftex
  @defaultparindent=0pt @parindent=0pt
@end iftex

@node    Top, , (dir), (dir)
@chapter Unsolved Problems
@section Odd Perfect Numbers
A number is said to be @i{perfect} if it is
the sum of its divisors.  For example, 6 is
perfect because
@tex $1+2+3 = 6$,
@end tex
@ifinfo
1+2+3 = 6,
@end ifinfo
and 1, 2, and 3 are the only numbers that divide
evenly into 6 (apart from 6 itself).

It has been shown that all even perfect numbers
have the form
@tex $$2^{p-1}(2^{p}-1)$$ where $p$ and $2^{p}-
1$
@end tex
@ifinfo
@center 2^(p-1) (2^p - 1)

where p and 2^p - 1
@end ifinfo
are both prime.

The existence of @i{odd} perfect numbers is an
open question.
@bye
```

```
Unsolved Problems
*****************

Odd Perfect Numbers
===================

A number is said to be perfect if it is the sum
of its divisors.  For example, 6 is perfect
because 1+2+3 = 6, and 1, 2, and 3 are the only
numbers that divide evenly into 6 (apart from 6
itself).

It has been shown that all even perfect numbers
have the form

                2^(p-1) (2^p - 1)

where p and 2^p - 1 are both prime.

The existence of odd perfect numbers is an open
question.
```

Figure 4: *Online documentation produced by MakeInfo*

3.2.10 Other Formats

There are a number of other macro packages available for TEX. Some of them are summarized below. The fact that they are not discussed more fully here (or listed below, for that matter) is not intended to reflect on the quality of the format. The formats discussed above are examples of the ways in which TEX can be extended. All of the formats below extend TEX in a way similar to one of the formats already mentioned. For any particular application, one of these macro packages might be a better choice than the formats discussed above.

EDMAC Provides support for typesetting critical editions of texts in a format similar to the Oxford Classical Texts with marginal line numbers and multiple series of footnotes and endnotes keyed by line number. EDMAC is available from the CTAN archives in the directory *macros/plain/contrib/edmac*.

INRSTEX Provides support for multilingual documents in French and English. INRSTEX is available from the CTAN archives in the directory *macros/inrstex.*

LamSTEX Extends AMSTEX with LATEX-like features and improved support for commutative diagrams. LamSTEX is available from the CTAN archives in *macros/lamstex.*

REVTEX Extends LATEX to provide support for typesetting articles for journals of the American Physical Society, the Optical Society of America, and the American Institute for Physics. REVTEX is available in the directory *macros/latex/contrib/revtex* of the CTAN archives.

TEXsis Provides facilities for typesetting articles, papers, and theses. It is particularly tuned for physics papers. TEXsis also provides support for other kinds of documents, such as letters and memos. It is based upon Plain TEX. TEXsis is available from the CTAN archives in the directory *macros/texsis.*

In addition to REVTEX and TEXsis, there are several other packages in the *macros* directory on CTAN that were designed for typesetting documents about physics: PHYSE, PHYZZX, and PSIZZLE.

TEX/Mathematica Supports interactive use of Mathematica on UNIX workstations running GNU emacs. Mathematica explorations can be annotated with TEX/LATEX, and Mathematica graphics can be incorporated into documents. TEX/Mathematica is available from the CTAN archives in the directory *macros/mathematica.*

ScriptTEX Supports typesetting screenplays in TEX. ScriptTEXmacro packages!ScriptTEX is available from the CTAN archives in the directory *macros/scripttex.*

VerTEX Supports typesetting articles for economic journals. VerTEX is available from the CTAN archives in the directory *macros/plain/contrib/vertex.*

3.3 Special-purpose Formats

In addition to the general-purpose packages discussed above, there are dozens, if not hundreds, of extensions to TEX that are designed for very specific tasks. Many of the extensions are LATEX style files; they provide styles for many academic journals, university theses, resumés, diagrams of various sorts, PostScript interfaces, linguistics, multinational language support, UNIX ‘man’ pages, program listings, and countless other tasks.

To give you a feel for the range of tasks that TEX can perform, I’ve selected a few packages to highlight the latitude of customization that is possible. Figure 5 shows the chemical structure of caffeine (a molecule dear to my heart) rendered with the ChemTEX package. The source is shown in Example 7. Another chemistry package, ChemStruct, was used to draw Figure 6. Its source is shown in Example 8. Taking TEX in another direction, the MusicTEX package was used to typeset the first two bars of Mozart’s K545 sonata in C-major in Figure 7. The MusicTEX input is shown in Example 9. Several more examples are presented in Chapter 7 where formats for typesetting non-English languages are described.

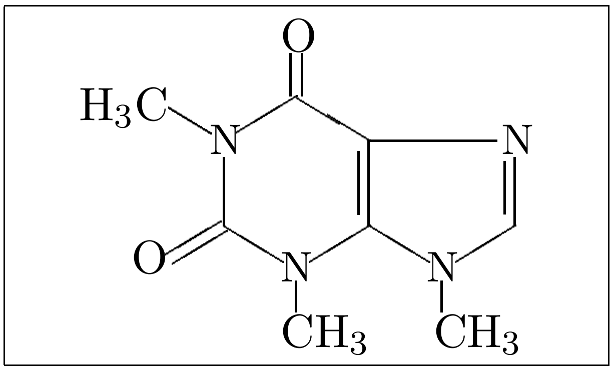


Figure 5: *Caffeine by ChemTEX*

Example 7: The ChemTEX Source for Caffeine

```
\initial
\len=4
\def\H{\hbox{\rm H}}
\def\C{\hbox{\rm C}}
\def\O{\hbox{\rm O}}

\[ \purine{$\H_3\C$}{$\O$}{$\C\H_3$}
{Q}{$\O$}{Q}{Q}{D}{$\C\H_3$}  \]
```

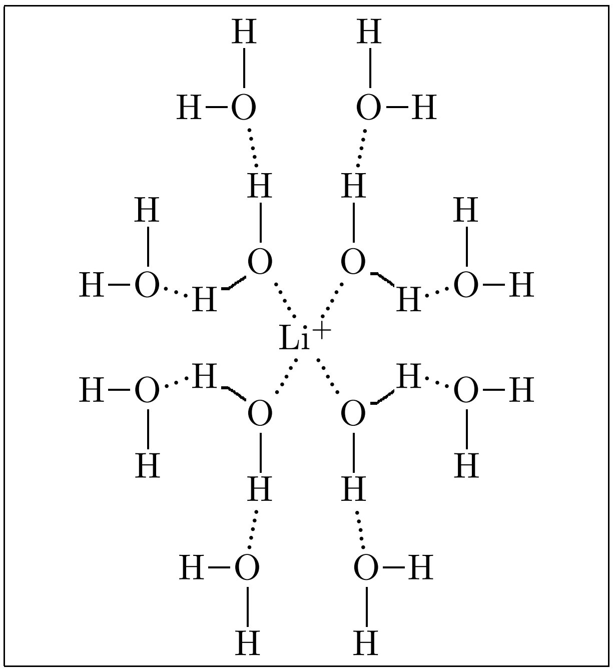


Figure 6: *A lithium cation rendered by ChemStruct*

Example 8: The ChemStruct Source for the Lithium Cation

```
\structure{\atom{~~Li$^+$}
\side{\nwbelow\atom{O}
\side{\nsingle\atom{H}\nnwbelow\atom{O}
\side{\wsingle\atom{H}}
\nsingle\atom{H}}\swsingle\atom{H}
\wnwbelow\atom{O}
\side{\wsingle\atom{H}}\nsingle\atom{H}}
\side{\nebelow\atom{O}
\side{\nsingle\atom{H}\nnebelow\atom{O}
\side{\esingle\atom{H}}
\nsingle\atom{H}}\sesingle\atom{H}
\enebelow\atom{O}
\side{\esingle\atom{H}}\nsingle\atom{H}}
\side{\swbelow\atom{O}
\side{\ssingle\atom{H}\sswbelow\atom{O}
\side{\wsingle\atom{H}}
\ssingle\atom{H}}\nwsingle\atom{H}
\wswbelow\atom{O}
\side{\wsingle\atom{H}}\ssingle\atom{H}}
\side{\sebelow\atom{O}
\side{\ssingle\atom{H}\ssebelow\atom{O}
\side{\esingle\atom{H}}
\ssingle\atom{H}}\nesingle\atom{H}
\esebelow\atom{O}
\side{\esingle\atom{H}}\ssingle\atom{H}}}
```

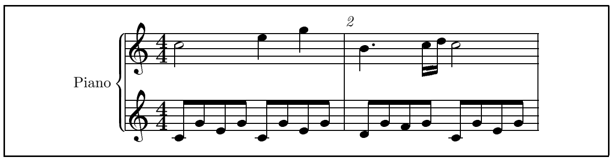


Figure 7: *A little Mozart. . .*

Example 9: The MusicTEX Source for Figure **??**

```
\begin{music}
\parindent 1cm
\def\nbinstruments{1}\relax
\def\instrumenti{Piano}%
\nbporteesi=2\relax
\generalmeter{\meterfrac{4}{4}}\relax
\debutextrait
\normal
\temps\Notes\ibu0f0\qh0{cge}
\tbu0\qh0g|\hl j\enotes
\temps\Notes\ibu0f0\qh0{cge}
\tbu0\qh0g|\ql l\sk\ql n\enotes
\barre
\Notes\ibu0f0\qh0{dgf}|
\qlp i\enotes
\notes\tbu0\qh0g|\ibbl1j3
\qb1j\tbl1\qb1k\enotes
\temps\Notes\ibu0f0\qh0{cge}
\tbu0\qh0g|\hl j\enotes
\finextrait
\end{music}
```

Another popular special-purpose application of TEX is the production of transparencies, also called foils or slides. There are a few different options for this application.

3.3.1 SliTEX

SliTEX is part of the standard LATEX distribution. Input to SliTEX consists of a main file and a slides file. Individual slides are composed in a `slide` environment.

SliTEX has provisions for black-and-white slides, color slides, and overlays. Unlike the other slide-making formats, which rely on `\special` printer commands[14] to incorporate color, SliTEX produces separate output pages for each color. For example, if you use red to highlight words on an otherwise black-and-white slide, SliTEX will produce two output pages for the slide: one with all the black text (excluding the words in red) and one with just the red words. Both of these pages will be printed in black. You must construct the colored slide by copying the pages onto colored transparencies and overlaying them. Producing slides with overlays in the same color is accomplished with a special 'invisible' color. This method of producing colored transparencies has been made obsolete by modern color printers. Of course, nothing prevents you from using the `\special` extensions of your DVI driver in SliTEX to produce colored transparencies directly on a color printer.

SliTEX has always been a separate macro package, distinct from and not 100% compatible with LATEX. With the introduction of LATEX2$_\varepsilon$, SliTEX is simply an extension package to the LATEX2$_\varepsilon$ core format. Support for a separate SliTEX format is being phased out.

3.3.2 FoilTEX

FoilTEX is an extension of LATEX for producing slides. The primary advantage of FoilTEX over SliTEX is that it is completely compatible with LATEX.[15] Note, however, that the defaults in many cases are not precisely the same as LATEX because of the radically different goal of FoilTEX.

FoilTEX provides support for running headers and footers, modified theorem environments for better mathematics in slides, support for AMS and PostScript fonts, and colors.

Color slides in FoilTEX are handled by DVI driver `\special` commands (most commonly *dvips* `\specials`). However, FoilTEX includes a number of new and extended features for better handling of color. See the section 'TEX in Color' later in this chapter for a detailed discussion of using color in TEX.

3.3.3 Seminar

The Seminar style is another alternative for producing slides and notes. Like FoilTEX, the Seminar style is designed to work on top of LATEX. (Seminar also works with AMSLATEX.) The Seminar style is designed to produce output for a PostScript printer. It isn't strictly necessary to produce PostScript output, but if you do not, many of Seminar's features will be unavailable to you.

Seminar provides for a mixture of portrait and landscape styles and can support color using either a color-separation technique (similar to SliTEX's method) or direct use of PostScript color. In either case, the PSTricks macro package is required. (Consult the section 'PSTricks' in Chapter 6, *Pictures and Figures*, for more information.)

The Seminar style has support for a number of interesting options (including two-up printing of slides), automatic resizing by changing a magnification parameter, instructions for converting your SliTEX slides, and several extensive demonstration files. Also included are explicit instructions for placing Encapsulated PostScript drawings in your slides.

[14] The `\special` mechanism is a way of passing arbitrary information through TEX to the DVI driver that will ultimately print the document.

[15] With the caveat that it still uses the Old Font Selection Scheme.

3.4 TEX in Color

With color printers and copiers becoming more common, the application of color, especially in transparencies, is more important than ever. Unfortunately, TEX knows *nothing* about color.

A little reflection about the design of TEX will make it clear why this is the case. TEX produces device-independent output. Even when color printers are as common as 'regular' printers, if that ever becomes the case, it will always be true that color is an inherently device-dependent attribute. It does not make sense for TEX to understand color. However, this does not prevent TEX from using color.

At the lowest level, all that is required to use color in TEX is some way of telling the printer 'start printing in <*color*> here.' This is easily accomplished with a `\special` command. In the discussion that follows, the *dvips* `\special` commands are used as concrete examples, but conceptually, any color printer can be used in this way.

3.4.1 Setting Up Color

Color support at the DVI driver level is provided by `\special` commands, but these are not typically convenient to enter directly into your document. Frequently, these commands are specified in terms of percentages of red, green, and blue (RGB color) or cyan, magenta, yellow, and black (CMYK color).

Higher-level support is provided by a collection of color control sequences. These sequences are loaded either by inputting the file *colordvi.tex* (in Plain TEX, for example) or using the *colordvi* style file (in LATEX).

dvips defines the colors in terms of 'crayon names.' If you need very precise control of the colors, you can adjust the precise mix of CMYK values in the file *colordvi.tex* after comparing the output of your printer with a standard scale (typically, the PANTONE scale). The following color names are standard in *dvips*.

Apricot	Emerald	OliveGreen	RubineRed
Aquamarine	ForestGreen	Orange	Salmon
Bittersweet	Fuchsia	OrangeRed	SeaGreen
Black	Goldenrod	Orchid	Sepia
Blue	Gray	Peach	SkyBlue
BlueGreen	Green	Periwinkle	SpringGreen
BlueViolet	GreenYellow	PineGreen	Tan
BrickRed	JungleGreen	Plum	TealBlue
Brown	Lavender	ProcessBlue	Thistle
BurntOrange	LimeGreen	Purple	Turquoise
CadetBlue	Magenta	RawSienna	Violet
CarnationPink	Mahogany	Red	VioletRed
Cerulean	Maroon	RedOrange	White
CornflowerBlue	Melon	RedViolet	WildStrawberry
Cyan	MidnightBlue	Rhodamine	Yellow
Dandelion	Mulberry	RoyalBlue	YellowGreen
DarkOrchid	NavyBlue	RoyalPurple	YellowOrange

3.4.2 Using Color

After *dvips* has loaded *colordvi*, typesetting text in color is simply a matter of using the appropriate color control sequence. For example, to typeset something in red, use the `\Red` control sequence in your document, like this:

```
\Red{something in red}
```

Alternatively, you can change the default text color with the `\textcolor` control sequences. To make default color for all text blue, enter:

```
\textBlue
```

To change the background color, use the `\background` macro. For example, to make the current and all future pages yellow, enter:

```
\background{Yellow}
```

You can enter a precise color by specifying it in terms of its CMYK components. The `\Color` and `\textColor` macros exist for this purpose. To typeset some text in a color that is 25% cyan, 35% magenta, 40% yellow, and 10% black,[16] enter:

```
\Color{.25 .35 .4 .1}{some text}
```

3.4.3 Now I've Got Color, but I Need Black and White!

If you have reason to print a colored TEX document on a black and white printer, you don't have to tear out all of the color commands. *dvips* includes a *blackdvi* file (analogous to *colordvi*—an input file or style file depending on your macro package), which translates all color commands into black and white.

Alternatively, 'good' implementations of PostScript in a black and white printer should translate all colors into shades of grey. This can be an inexpensive way to preview a color document. Most screen previewers simply ignore color commands so they print in black and white even if the document is colored.

3.4.4 Color Under LATEX2$_\varepsilon$

At the time of this writing, the LATEX2$_\varepsilon$ team has not officially adopted a standard for using color. However, it is likely to follow a slightly different model than the one described above. The final design should provide color selection commands that are device-independent at the DVI driver level (in other words, the color commands will not insert device-specific commands, like snippets of PostScript, into the DVI files).

[16] I made these numbers up. I take no responsibility for the artistic merits (or lack thereof) of the resulting color.

3.4.5 Color Is Subtle

Color commands implemented as `\special` commands may introduce occasional problems. For example, if TEX introduces a page break in a paragraph that you have typeset in yellow (`\Yellow{This is a long paragraph...}`), the resulting output may print the page footer (and even the header) in yellow, although that was not intended.

Circumventing these problems may require careful use of color commands in front of text that you want to appear black. For example, in Plain TEX the difficulty described above can be avoided by specifying that the page number should be printed this way:

```
\footline{\Black{\hss\tenrm\folio\hss}}
```

This definition guarantees that the page number will be set in black, and because it is a local color change, colored text can flow across the page around it.

You may want to make sure that other typographic elements are printed in the current global color (which may vary). *dvips* provides a local color macro called `\globalColor` for that purpose. Every time the text color is changed globally (with a `\textColor` command), `\globalColor` is redefined to print text in that color.

3.4.6 Further Reading

Read the documentation for your DVI driver carefully with respect to color. Because it is device-dependent, there is a lot of room for interpretation, and it may not always be obvious why some things appear the way they do. And DVI drivers are free to implement color in any way they choose.