UNIX Text Processing

New Lay-flat Binding

Learning

GNU Emacs

A NUTSHELL HANDBOOK

Debra Cameron and Bill Rosenblatt

O'Reilly & Associates, Inc.

Learning GNU Emacs
by Debra Cameron and Bill Rosenblatt

Editor: Mike Loukides

Printing History:

October 1991: First Edition.

April 1992: Minor corrections.

This book is printed on acid-free paper with 50% recycled content, 10-15% post-consumer waste. O'Reilly & Associates is committed to using paper with the highest recycled content available consistent with high quality.

ISBN: 0-937175-84-6 [4/93]

Preface

Why Read This Book?
Which Emacs is Which?
GNU Emacs and the Free Software Foundation
An Approach to Learning Emacs
Conventions Used in This Handbook
Acknowledgments

Emacs is the most powerful text editor in the UNIX world today. Unlike most other editors (in particular, unlike the standard UNIX editor, **vi**), Emacs is a complete working environment. No matter what you do, you can start Emacs in the morning, work all day and all night, and never leave it: you can use it to edit, rename, delete, and copy files; to compile programs; to do interactive work with the UNIX shell; and so on. Before window systems like X became popular, Emacs often served as a complete windowing system of its own. All you needed was an Ann Arbor terminal with a 64-line display, and you could live within Emacs forever. Emacs is also infinitely flexible; you can write your own commands, change the keys that are associated with commands, and (if you are willing to take the time) do just about anything you want.

Why Read This Book?

Because it does so much, Emacs has a reputation for being extremely complicated. To date, most of the Emacs manuals that have been published have been comprehensive reference manuals rather than how-to books designed for people

who are new to Emacs. That's the reason for this book: to teach you how to learn Emacs from the ground up, covering first the basics and then some of the more advanced features.

In this book, we have tried to reach as broad an audience as possible: from the administrative assistant who only needs to use Emacs to write mail messages and office memos, to the professional writer who needs to write complex documents full of formatting codes, to the advanced programmer who would like to use Emacs to format source code. No matter what you do with Emacs, you will find that it's easy to learn; after one or two sessions, you'll know the basics of editing any file. After you learn the basics, you can go on to learn about more advanced topics, which provide the real benefits of using Emacs. These include:

- Using multiple windows and buffers so you can work on more files at once.
- Customizing your keyboard commands.
- Tailoring Emacs to fit your work style using variables.
- Making Emacs your work environment where you can do all your everyday tasks, such as reading mail, compiling programs, and issuing shell commands.
- Creating macros to streamline repetitive tasks.
- Using Emacs to support programming in many languages (including C, LISP, and FORTRAN).
- Formatting files for documentation environments such as TeX and **troff**.
- Using word abbreviations to avoid spelling out long phrases or to correct common misspellings.

Of course, many of the topics may not apply to you; some topics may be appropriate for a second reading, but not for the first. Towards the end of the preface, we'll sketch several different ways to approach the book, depending on your interests and experience.

We do make a few assumptions. We assume that you're basically familiar with UNIX (or, if not UNIX, whatever operating system you're using). In particular, you should know what files and directories are, how they are named, and the basic things that you can do with them (copy, delete, rename; under UNIX, these are done with the **cp**, **rm**, and **mv** commands). If you're completely new to UNIX, we recommend that you read the Nutshell Handbook, *Learning the UNIX Operating System*, by Grace Todino and John Strang.

Which Emacs is Which?

Many versions of Emacs are available, offering a wide range of features. For UNIX, the most important versions of Emacs are GNU Emacs, which we cover in this book; Unipress Emacs (also called Gosling Emacs, after its author); and CCA Emacs. Some versions (Freemacs, MicroEmacs, Epsilon) run on personal computers under MS-DOS. Some other versions offer a very limited set of features, but in return are relatively sparing of memory and CPU cycles; the "mainline" Emacs implementations tend to be hard on computers, particularly if many users share a central computer. Some Emacs-like editors (Epoch) are very closely integrated with the X Window System. Some versions are "free" (which may mean any of several different things), and some are rather expensive commercial products. Some other programs (like FrameMaker) are not Emacs implementations in any way but have a similar set of keyboard commands.

As you've probably gathered, Emacs isn't so much an "editor" as a family of editors. There are a lot of similarities; all of these editors allow to you to mix text and editor commands freely; most of these editors support work in multiple windows; they all provide some kind of support for working with special file types (like C programs, TeX files, etc.); and most of them are heavily customizable, usually by programming in some dialect of LISP.

Given all this confusing diversity, which version of Emacs have we described? This book covers GNU Emacs, Version 18.[1] Since its appearance, it has become by far the most popular Emacs implementation in the UNIX world, and there's no reason to believe that this will change. It is also the most powerful and the most flexible of these extremely powerful and flexible editors. If you know GNU Emacs, you will be able to adapt to any other Emacs implementation with no trouble; it's not so easy to go in the other direction. Given these facts, our only logical choice was to focus the book on GNU Emacs.

However, this book isn't limited to GNU Emacs users. Because of the "family relations" between different Emacs implementations, this book should help you get started with any Emacs editor. The basic keyboard commands change little from one editor to another—you'll find that **C-n** (CONTROL-n) almost always means "move to the next line." Emacs editors tend to differ in the more advanced commands and features—but if you are using these more advanced facilities, and you aren't using GNU Emacs, you should most certainly consider making the switch.

---

[1] Version 19 (with some significant extensions) wasn't available at this writing. The most significant addition to Version 19 is extended X Window System support.

GNU Emacs and the Free Software Foundation

You don't need to know history to use GNU Emacs, but its sources are an interesting part of recent computer history. The Free Software Foundation, which maintains and distributes GNU Emacs, has become an important part of computer culture.

A long time ago (1975), Richard Stallman at MIT wrote the first Emacs editor. According to the folklore, the original Emacs editor was a set of macros for TECO, an almost incomprehensible and now obsolete line editor. The name Emacs stands for "Editing Macros." Tradition also has it that Emacs is a pun on the name of a favorite ice cream store. Much has happened since 1975. TECO has slipped into deserved obscurity, and Emacs has been rewritten as an independent program. Several commercial versions of Emacs have appeared, of which "Unipress Emacs," also known as "Gosling Emacs," and "CCA Emacs" were the most important. For several years, these commercial implementations were the Emacs editors you were most likely to run across outside of the academic world.

Stallman's Emacs became prominent with the birth of the Free Software Foundation and the GNU Project. GNU stands for "GNU's Not UNIX" and refers to a complete UNIX-like operating system that Stallman and his associates are building. Stallman founded the Free Software Foundation to guarantee that some software would always remain "free." Free does not necessarily mean cheap (you may have to pay a fee to cover the cost of distribution); it most definitely means liberated from restrictions about how it can be used.

To understand what "free" means, we have to look at how software is typically distributed. Most commercial software comes with a highly restrictive license. You have to pay to use the program; you probably have to pay separately for each computer that runs the program; you may have to pay more money every year to continue using the program; in some cases, you even have to pay by the minute. You are most definitely *not* allowed to give the program to your friends, and you will probably *never* (unless you are very wealthy) see the program's source code. If a commercial program is broken or doesn't have some feature that you need, you are completely at the mercy of the company you bought it from. And they may well decide to ignore you.

GNU Emacs has none of these limitations. If you can find someone to give it to you (and you usually can), you can get it for free. It is available from many pub-

lic archives, including UUNET and several "anonymous FTP" sites (we'll discuss this in Appendix A). It is always distributed with the source code, so if you're a programmer, you can add your own features and fix your own bugs.[2] You can give copies to your friends. You will never be asked to pay for the right to use it. About the only thing you cannot do is impose further restrictions on how Emacs is used: i.e., if you give a copy of Emacs away, or make some improvements to it, you cannot suddenly start charging licensing fees. GNU Emacs is free, and will remain free. Your rights and responsibilities as a user are described precisely in the "General Public License," which we have published in Appendix F.

The Free Software Foundation was created precisely to distribute programs under terms which encourage you to share software, rather than to "hoard" it. The General Public License is designed to prevent an unfortunately common practice: companies taking public-domain code, making a few modifications and bug fixes, and then copyrighting their modified version. Once this has happened, the program has essentially become "private property" and disappeared from the public domain. Stallman formed the Foundation because he finds this practice abhorrent. As he explains in the GNU Manifesto, "I cannot in good conscience sign a nondisclosure agreement or a software license agreement... So that I can continue to use computers without dishonor, I have decided to put together a sufficient body of free software so that I will be able to get along without any software that is not free." Elsewhere in the manifesto, Stallman calls sharing software the "fundamental act of friendship among programmers." Their software is free because it *can* be shared, and will *always* be shareable–without restriction.[3]

Since GNU Emacs was first released, many other pieces of the GNU operating system have fallen into place: C and C++ compilers (*gcc* and *g++*), a very powerful debugger (*gdb*), substitutes for **lex** and **yacc** (called *flex* and *bison*, respectively), a UNIX shell (*bash*, which stands for "Bourne-Again Shell"), and many other programs and libraries. Many pre-existing facilities, like the RCS source code control system, have been placed under the Free Software Foundation's protection. Among other things, the Foundation is currently developing a UNIX-like kernel. When this is finished, they will have a complete modern operating system. Given the quality of their software, the GNU kernel will certainly be competitive with, if not better than, any of the commercial UNIX implementations that are now available.

---

[2] Fortunately, there are very few bugs—indeed, the Free Software Foundation produces much better code than most for-profit companies, and they are very responsive to bug reports. But if you do find a bug, and can't wait for them to fix it, you can do it yourself.

[3] Free Software Foundation programs, like Emacs, are often distributed with commercial systems. Even in these cases, the General Public License guarantees your right to use and give away their programs without restriction. Of course, the license does not apply to other proprietary software with which GNU tools have been shipped.

An Approach to Learning Emacs

Just as there are many versions of Emacs, there are many types of Emacs users. This book is designed to get you started with Emacs as quickly as possible, whether you are an experienced computer user or a novice. The first two chapters give you the basics you need to know, and the rest of the book builds on these. After the first two chapters, you don't have to read the rest consecutively; you can skip to the topics that interest you. Additionally, the book is designed to give you just the level of "hand-holding" you want; you can either read the book in detail or skim it, looking for tables of commands and examples.

Here are some reading paths you could take:

If	Read
You are an administrative user	Preface, Chapters 1-3, 13.
You are a casual user	Preface, Chapters 1-3, 13.
You are a programmer	Preface, Chapters 1-4, 8-11.
You are a writer or production person	Preface, Chapters 1-4, 6-7, 13.
You want to customize Emacs	Chapter 9, possibly 11.
You want to use mail in Emacs	Chapter 5.
You use UNIX commands in Emacs	Chapter 5.

These reading paths are only offered as a guideline. Emacs is one gigantic, functionally rich editor. We've divided it up into digestible bites for you, so you don't have to be put off by its size and scope. The best way to learn Emacs is incrementally; learn a little now, then learn more features as you get curious about them or find you need to do something and don't know how to do it in Emacs. Emacs probably already does it; if it doesn't, you can learn how to write a LISP function to add it to Emacs (see Chapter 11 for details). The online help system in GNU Emacs is an excellent place to learn about new features on the fly; how to use online help is discussed in Chapter 1, and in more detail in Chapter 13.

Here's a list of some features you might want to learn about on a rainy day:

- Word abbreviation mode (Chapter 3).
- How to use macros (Chapter 8).
- How to map your function keys to Emacs commands (Chapter 9).
- How to issue (and edit) shell commands from Emacs (Chapter 5).
- How to use multiple windows (Chapter 4).
- How to make simple drawings in picture mode (Chapter 6).

Finally, if you insist on reading through the book from beginning to end, here's a quick summary of what's in each chapter:

Chapter 1, *Emacs Basics*, tells you how to start Emacs and how to work with files. It also provides a quick introduction to Emacs' online help system.

Chapter 2, *Editing Files*, explains editing with Emacs, including commands for moving around, copying and pasting text, and undoing changes.

Chapter 3, *Search and Replace Operations*, covers more editing features, including search and replace, word abbreviation mode, and checking spelling.

Chapter 4, *Using Buffers and Windows*, describes how to use multiple buffers and windows.

Chapter 5, *Emacs as a Work Environment*, talks about Emacs as a work environment, where you can do everything you can do at the shell prompt, including reading and writing mail, issuing shell commands, and working with directories.

Chapter 6, *Simple Text Formatting and Specialized Editing*, covers the basic text formatting (such as indentation and centering) that you can do in Emacs as well as some of the more rarified features like picture mode and outline mode.

Chapter 7, *Using Emacs with UNIX Text Formatters*, describes Emacs support for **troff** (and its relatives), TeX, LaTeX, and Scribe.

Chapter 8, *Writing Macros*, discusses using macros to eliminate repetitive tasks.

Chapter 9, *Customizing Emacs*, explains how to customize Emacs according to your preferences: setting up your terminal, customizing your keyboard commands, tailoring your editing environment, and loading Emacs packages for extra functionality.

Chapter 10, *Emacs for Programmers*, covers Emacs as a programming environment, including editing support for C, LISP, FORTRAN, and other languages, and Emacs' interface to compilers and the UNIX **make** utility.

Chapter 11, *Emacs LISP Programming*, describes the basics of Emacs LISP, the language you can use to customize Emacs even further than with the techniques discussed in Chapter 9.

Chapter 12, *Emacs for the X Window System*, discusses Emacs' interface to the X Window System, allowing you to use a mouse and pop-up menus if you use a graphics workstation.

Chapter 13, *Online Help*, describes Emacs' rich, comprehensive online help facilities.

Appendix A, *How to Get Emacs*, tells how you can get GNU Emacs as well as a few other versions of Emacs.

Appendix B, *Making Emacs Work the Way You Think It Should*, tells you how to ensure that your Emacs behaves as described in this book.

Appendix C, *Emacs Variables*, lists many important Emacs variables, including all the variables mentioned in this book.

Appendix D, *Emacs LISP Packages*, lists most of the LISP packages that are available as of this publication.

Appendix E, *Bugs and Bug Fixes*, tells you how (and when) to report bugs that you find in Emacs.

Appendix F, *Important Documents*, reprints the text of the General Public License, which gives the rules under which GNU Emacs is distributed.

Appendix G, *Give and It Shall Be Given*, tells you how you can support the Free Software Foundation in their efforts to create more quality software.

What We Haven't Included

GNU Emacs is an extremely large and powerful editor; in this book, we've only been able to give you a sample of what it does. Many features have been left out, and more features are added all the time.

However, there are some particular things that are missing:

- Compatibility modes: GNU Emacs provides compatibility modes for the UNIX **vi** editor, the VAX/VMS EDT editor, and the Gosling Emacs editor. We've left this out. In our experience "compatibility modes" tend to be poor emulations of the "real thing." If you really want to use **vi** or EDT, do so. You're better off getting to know Emacs on its own terms, rather than pretending it is something else.
- Many programming language modes: In this book, we discuss the editing modes for C, LISP, and FORTRAN. There are many modes for other languages; some work better than others, and new modes are added frequently. There's no way we could discuss everything.
- Advanced LISP programming: GNU Emacs incorporates a complete LISP interpreter. We give a very basic and brief introduction to Emacs LISP; Chapter 11 should be enough to get you started, but really only scratches the surface. The Free Software Foundation publishes a complete *Emacs LISP Reference Manual*; if there is sufficient demand, O'Reilly and Associates will consider publishing a book on advanced Emacs LISP programming.

- Porting, Debugging, and Installation: This book doesn't describe the problems that can arise while porting GNU Emacs to other systems. We believe that such a book is badly needed, but it's far beyond the scope of the present volume. Let us know if you'd find this useful, and we'll consider writing it.
- Games and Amusements: GNU Emacs includes an eclectic bunch of games and amusements, including the ability to pipe random quotations from Zippy the Pinhead into the famous "Eliza" pseudo-psychoanalyst. (Type **ESC-x psychoanalyze-pinhead RETURN** sometime and see what happens.) Alas, we had to draw the line somewhere.

Conventions Used in This Handbook

This section covers the conventions used in this book.

Emacs Commands

Emacs commands consist of a modifier, such as CTRL (CONTROL) or ESC (ESCAPE), followed by one or two characters. Commands shown in this book abbreviate CTRL to C:

C-G Hold down the **CTRL** key and press **G**.

To complete a command you may need to press a carriage return:

RETURN Press the **RETURN** key.
This key may be labelled **ENTER** on your keyboard.

Most Emacs manuals refer to the META key in addition to the CTRL key. Since most keyboards don't have a META key, this book refers to ESC instead of META:

ESC-x Press **ESC**, *release it*,[4] then press **x**.

If your keyboard does have a META key, it works like the CTRL key described above, that is, you hold down the META key and press the desired key, such as G. Since you can continue to hold down the META key for repeated keystroke sequences, META keys do have an advantage. ESC tends to be less convenient for

[4] We emphasize this because pressing ESC twice (**ESC ESC**) or holding it down a second too long so that it repeats gives you an error message; see the Problem Checklist at the end of Chapter 2, *Editing Files*, for more information.

repeated keystroke sequences. In general, if you have a META key on your keyboard, you will probably prefer to use it instead of ESC.

A few mouse commands (used only with the X Window System, discussed in Chapter 12) use the SHIFT key as a modifier, often in combination with the CTRL key. This is abbreviated as:

S-right Press the right mouse button in combination with the SHIFT key.

C-S-right Press the right mouse button in combination with the SHIFT and CTRL keys.

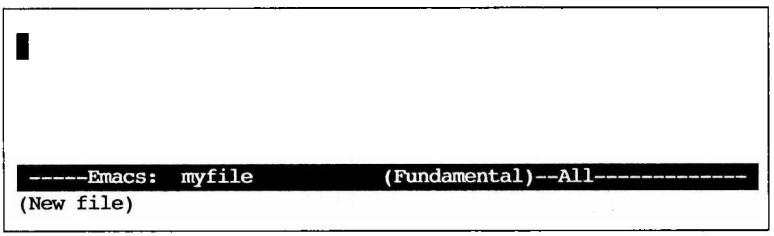
All Emacs commands, even the simplest ones, have a "full name": for example, **forward-word** is equivalent to the keystrokes **ESC-f** and **forward-char** is equivalent to **C-f**. Many commands only have "full names"; there are no corresponding keystrokes.

When we're discussing a command, we'll give both its full name and the keystrokes (if any) that you can type to invoke it. We will assume that you are using the "default bindings" (the default assignment of keystrokes to Emacs commands).

Examples

Throughout the book, you'll find keystrokes to type, followed by an Emacs screen showing the results.

Type: **emacs** *myfile*



Type **emacs** with a filename at the UNIX prompt to start an Emacs session.

As shown in the Keystrokes column, the word **emacs** is in bold constant width font, indicating that this is exactly what you type. *myfile* is shown in italics because you could substitute any filename you choose and need not type exactly what you see here. The cursor (the point at which you are currently editing) is shown in "reverse video," as is the Emacs "mode line" at the bottom of the screen.

Towards the end of the book, when we're discussing programming modes, customization, and LISP programming, showing a simulated Emacs screen becomes rather unwieldly. Therefore, we'll eventually stop using displays like the one above; instead, we'll only show one or two lines of text. If it's relevant, we'll show the cursor's position in reverse video:

```
/* This is a C comment */
```

Font Usage

This book uses the following font conventions:

- Emacs keystrokes, command names, and variables are shown in **boldface** type.
- Filenames are shown in *italic* type.
- LISP code, C code, and other excerpts from programs are shown in `constant width` type.
- Dummy parameters that you replace with an actual value are shown in *italic* type. (If they appear within a program, they are shown in *`constant width italic`* type.)
- Buffer names are shown in `constant width` type.
- UNIX commands are shown in **boldface** type.

To find a group of commands quickly, look for tables in each section that summarize commands. These tables are formatted like this:

Sample Command Table

Keystrokes	Command Name	Action
C-n	next-line	Move to the next line.
(none)	yow	Print wisdom from the Pinhead in the minibuffer.

The first column shows the default key binding for the command, the second shows the command's full name, and the third describes what the command does. Many commands aren't bound to particular keystrokes; for example, the **yow** command. In this case, we'll put "(none)" in the first column. This doesn't mean

you can't use the command; just type **ESC-x**, followed by the command's full name, and type a **RETURN**. (Try **ESC-x yow** sometime.)

Acknowledgments

Debra Cameron: I'd like to thank Peter Mui for hatching the original concept of this book and keeping it alive, and Chris Genly, for lending a willing hand with his Emacs and LISP expertise in the early days of writing this book, Duffy Craven for infecting me with his boundless enthusiasm for Emacs and teaching me lots of helpful tricks, Ted Stefanik for giving us an idea of what programmers and other techies would like to see in this book, and all the people who supported me and helped me to get this book done in pretty outrageous circumstances: my wonderful husband Jim (and Meg and David, too), Barbra Gibson, Betty Avins, Barbara Burkhardt, Cary Rodgers, and Jeanie O'Hara.

Bill Rosenblatt: I would like to thank the following people: Professor Richard Martin (Princeton Classics Department), for planting the seed in me that eventually turned writing from a chore to a pleasure; Intermetrics, Inc., for giving me little enough to do that I could fritter away my workdays delving into GNU Emacs; Hal Stern, for getting me this gig; Sandy Wise, for his help with the X Window System chapter; Jessica Lustig, for her love and support; and most importantly, my grad-school housemates for putting up with a tied-up phone line at all hours of the day and night.

We'd also like to thank our technical reviewers who gave of their time and expertise: Andy Oram, Rick Farris (and his friend Rock Kent), Eileen Kramer, Linda Mui, and Chris Genly. In addition to their careful review on technical matters, Andy Oram gave us the perspective of an accomplished writer, Rick Farris kept us smiling, Chris Genly wisely read the last part first, and Linda Mui, Eileen Kramer, Bruce Barnett, and Judy Loukides gave us insight about how new users would view the book.

Most of all, we'd like to thank our editor, Mike Loukides, who—in addition to artful editing on every level—made sure that the book was as complete as possible, writing whole sections himself at times to make sure the job got done. And special thanks to Eileen Kramer and all the great folks at ORA who copyedited, produced, indexed, and made this book a reality.

Table of Contents

Page

Preface xvi

Why Read This Book? xvi
Which Emacs is Which? xviii
GNU Emacs and the Free Software Foundation xix
An Approach to Learning Emacs xxi
  What We Haven't Included xxiii
Conventions Used in This Handbook xxiv
  Emacs Commands xxiv
  Examples xxv
  Font Usage xxvi
Acknowledgments xxvii

Chapter 1 Emacs Basics 1

Introducing Emacs! 1
Understanding Files and Buffers 3
A Word about Modes 4
Starting Emacs 6
About the Emacs Screen 7
Emacs Commands 8
Reading a File 9
  Letting Emacs Fill in the Blanks 11
  Inserting and Appending Files 12
  How Emacs Chooses a Default Directory 13
Saving Files 13
Leaving Emacs 14
  Temporarily Suspending Emacs 14
Customizing Emacs and its Pitfalls 15
Getting Help 17
Summary 19
  Problem Checklist 20

Chapter 2 Editing Files 22

Text Mode and Fill Mode 23
What Happens Without Fill Mode 23
Moving the Cursor 24
Repeating Commands 25
Other Ways to Move the Cursor 27
Moving a Screen (or More) at a Time 29
Redisplaying the Screen 30
Deleting Text 31
Recovering What You've Deleted 32
Marking Text to Delete, Move, or Copy 35
Copying Text 38
More about the Kill Ring 39
Reformatting Paragraphs 40
Stopping and Undoing Commands 41
Stopping Commands 42
Undoing Changes 42
Backup Files 43
Editing Tricks and Shortcuts 44
Fixing Transpositions 44
Capitalization 45
Typing over Old Text with Overwrite Mode 47
Problem Checklist 47

Chapter 3 Search and Replace Operations 49

Different Kinds of Searches 49
Incremental Search 51
Simple Searches 54
Word Search 55
Search and Replace 55
Simple Search and Replace Operations 56
Query-replace 57
Recursive Editing 59
Are Emacs Searches Case-sensitive? 60
Regular Expressions for Search and Replacement Operations 61
Checking Spelling 63
Word Abbreviation Mode 67
Trying Word Abbreviations for One Session 69
Making Word Abbreviations Part of Your Startup 70

Deleting a Word Abbreviation 70
Disabling Word Abbreviations 71
Abbreviations and Capitalization 72

Chapter 4 Using Buffers and Windows 74

Files, Buffers, and Windows 74
Working with Multiple Buffers 76
Saving Multiple Buffers 78
Deleting Buffers 78
Renaming Buffers 79
Read-only Buffers 80
Getting a List of Buffers 80
Working with the Buffer List 82
Working with Windows 85
Creating Horizontal Windows 86
Creating Vertical or Side-by-side Windows 87
Moving Between Windows 88
Getting Rid of Windows 90
Growing Windows and Shrinking Them 90
Shortcut Commands for Working with Other Windows 92
Comparing Files Between Windows 92
Displaying Buffers from the Buffer List 94

Chapter 5 Emacs as a Work Environment 95

Working with Mail 96
Sending Mail from within Emacs 96
Executing UNIX Commands in Shell Windows 118
Using Shell Mode 122
Working with Directories 128
Getting into Dired 128
Deleting Files with Dired 130
Copying and Renaming Files with Dired 131
Printing from Emacs 133
Reading Man Pages from Emacs 134
Using Your Emacs Work Environment 135