PREFACE

The hungry sheep look up, and are not fed . . .

—John Milton, *Lycidas*

Copyright © 1993 O'Reilly and Associates, Inc. All rights reserved. Permission is granted to copy this document for personal use only. This copyright notice must be retained in all copies.

This book is about *imake*, a UNIX tool that helps you write portable programs.

Most program development under UNIX is done with *make*, using a *Makefile* to direct the build and install processes. But Makefiles aren't portable, and it's often difficult to rewrite them by hand to accommodate machine dependencies for different systems. *imake* provides an alternative based on a simple idea: describe the dependencies for various systems in a set of configuration files and let *imake* generate the *Makefile* for you after selecting the dependencies appropriate for your machine. This frees you from writing and rewriting Makefiles by hand and helps you produce portable software that can be built and installed easily on any of the systems described in the configuration files.

imake has been used successfully to configure software such as the X Window System, the Kerberos authentication system, and the Khoros software development environment. X in particular is a large project that, despite its size and complexity, is remarkable in its portability. Much of this portability is due to the use of *imake*, which is thus arguably one of the reasons X has been so successful.

Nevertheless, despite the fact that it's freely available and runs on a variety of systems, *imake* isn't as widely exploited as it could be. There are at least two reasons for this:

- *imake* itself is a simple program, but the configuration files it uses sometimes are not. In particular, most people first encounter *imake* through the configuration files distributed with X. The X files are powerful and flexible, but they're also complicated and forbidding, and don't provide a very accessible entry point into the world of *imake*.
- *imake* hasn't been well documented. Normally when you don't understand how to use a program, you turn to the documentation. But for *imake* there hasn't been much available, which compounds the difficulty of learning it. Consequently, *imake* remains a mystery, leaving potential *imake* users in the predicament of Milton's sheep—seeking help in vain.

I freely acknowledge that *imake* can be difficult to make sense of. But it doesn't *need* to be so, nor should you have to become an initiate into the Eleusinian mysteries to be granted an understanding of how *imake* works.

Why Read This Handbook?

The goal of this book is to show how *imake* can help you write portable software and to make it easier for you to use *imake* on a daily basis. Then you'll be able to see it as a tool to be used, not a stumbling block to trip over.

If you don't know how to use *imake* at all, this book will teach you. If you already use *imake*, you'll learn how to use it more effectively. The book provides assistance on a number of levels, and you'll find it useful if you're in any of the situations below:

- You wonder what *imake* is and how it works.

— Copyright © 1993 O'Reilly & Associates. All rights reserved. —

- You couldn't care less what *imake* is or how it works. (As in, "I just got this program off the net. What do I do with this *Imakefile* thing? I'm not interested in *imake*, I just want to get the program built!")
- You're curious about the relationship between an *Imakefile* and a *Makefile*.
- You're faced with the task of using *imake* for the first time and are finding it less than obvious.
- You're trying to use *imake* but you need help diagnosing problems that occur. For example: "I just generated my *Makefile*, but *make* says it contains a syntax error; what do I do now?"
- You're tired of editing your Makefiles every time you move your programs from one machine to another.
- You've inherited projects that were developed using *imake* and you need to understand how they're configured so you can maintain them.
- You've been able to use *imake* to configure, build, and install the X Window System on your workstation using the instructions provided with the X source distribution, but it all seemed like magic. You want to better understand what goes on during that process.
- You're planning to write X-based software. It's best to do this using an *Imakefile* (since X itself is *imake*-configured), and you'll be required to provide one anyway if you plan to submit your software to the X Consortium for inclusion in the X *contrib* distribution.
- You admire the portability of X and want to achieve the same for your own software, but you've found it difficult to use *imake* without a copy of X nearby. You suspect it's possible for *imake* to stand on its own, but you're not sure how.
- You're looking for a general purpose tool for long-term software development and are considering using *imake* to that end.

Scope of This Handbook

This book is divided into three parts. Part I provides an overview of *imake* and how to use it, a basic description of the operation of configuration files, and how to write and troubleshoot Imakefiles. Part II describes how to write your own configuration files. Part III comprises appendices for your reference. Each chapter and appendix is described briefly below: use this information to navigate to those of most interest to you.

Part I

Chapter 1, *Introduction*, describes what *imake* is and what it does. It also discusses why *make* is inadequate for achieving software portability. Read this to find out what problems *imake* attempts to solve.

Chapter 2, *A Tour of imake*, is an *imake* tutorial. In this chapter, I assume that you're not particularly interested in details about how *imake* works, you just want to know how to use it to do specific things: how to write a simple *Imakefile* to specify programs you want to build, how to generate the *Makefile* from the *Imakefile*, etc. Instead of starting from "first principles," you use *imake* to run through some basic exercises to get a feel for what it does and how to make it do what you want.

Chapter 3, *Understanding Configuration Files*, describes the principles governing the design of configuration files that you need to know to understand their structure and content—what's in them and how they work together. You don't need to read this chapter if you're a casual *imake* user, but you should if you want to employ *imake* more effectively or if you plan on writing your own configuration files. This chapter is the point at which I begin to assume you want to understand *imake*'s workings in more detail—not just what it

— Copyright © 1993 O'Reilly & Associates. All rights reserved. —

does, but how and why.

Chapter 4, *The X11 Configuration Files*, discusses the configuration files from the X Window System, relating them to the general principles described in Chapter 3 and pointing out some of their unique features. This isn't an X book, but the X11 configuration files are the most well known instance of the use of *imake* to date. As such, they provide a fertile source of examples and discussion. It would be unwise not to take advantage of the lessons they provide.

Chapter 5, *Writing Imakefiles*, describes how to write Imakefiles for programs configured with the X11 configuration files. This chapter provides simple examples that require little or no knowledge of the X11 files, as well as detailed discussion to increase your practical understanding of how the X11 files work.

Chapter 6, *Imakefile Troubleshooting*, discusses things that can go wrong when you write Imakefiles and how to fix them.

Part II

Chapter 7, *A Closer Look at Makefile Generation*, examines the process by which *imake* builds Makefiles, including a discussion of how `Makefile` and `Makefiles` target entries work.

Chapter 8, *A Configuration Starter Project*, shows how to convert the X11 configuration files into a starter project you can use as a jumping-off point for developing new projects or new sets of configuration files.

Chapter 9, *Coordinating Sets of Configuration Files*, discusses the problems that arise in a world populated by multiple sets of configuration files and how to solve the problems so you can manage those files easily.

Chapter 10, *Introduction to Configuration File Writing*, shows by example how to write a set of configuration files. It describes in general the various ways you can modify the contents of configuration files.

Chapter 11, *Writing Rule Macros*, continues the discussion begun in Chapter 10, focusing on the design and implementation of *imake* rules.

Chapter 12, *Configuration Problems & Solutions*, discusses several configuration problems and shows how to solve them using the principles and techniques described in Chapters 10 and 11.

Chapter 13, *Troubleshooting Configuration Files*, discusses things that can go wrong when you write configuration files and how to fix them. It complements Chapter 6, *Imakefile Troubleshooting*.

Chapter 14, *Designing Extensible Configuration Files*, discusses how to design configuration files to be shared easily among projects, while allowing individual projects to specify their own particular requirements by extending or overriding the information in the shared files. This extensible architecture reduces the need to write new configuration files for a project when existing files don't quite match a project's configuration requirements.

Chapter 15, *Creating Extensible Configuration Files*, describes a procedure you can use to convert existing project specific configuration files to the extensible architecture.

Chapter 16, *Using Extensible Configuration Files*, shows how to develop new projects that take advantage of the flexibility afforded by the extensible architecture.

Part III

Appendix A, *Obtaining Configuration Software*, describes how to get the software described in this book. *imake* is owned and copyrighted by MIT but may be freely redistributed. So although *imake* isn't usually included as part of the software distributed with the UNIX operating system, anyone with FTP or GOPHER access on the Internet can get it. *imake* is also available by electronic mail.

— Copyright © 1993 O'Reilly & Associates. All rights reserved. —

Appendix B, *Installing Configuration Software*, discusses how to build and install the software described in this book.

Appendix C, *Configuration Programs: A Quick Reference*, documents the software described in this book in "abbreviated manpage" format.

Appendix D, *Generating Makefiles: A Quick Reference*, briefly describes how to build a *Makefile* from an *Imakefile*.

Appendix E, *Writing Imakefiles: A Quick Reference*, briefly describes how to write an *Imakefile*.

Appendix F, *Writing Configuration Files: A Quick Reference*, briefly describes how to write configuration files.

Appendix G, *Basics of make and cpp*, provides brief overviews of *make* and *cpp*. Since *imake* produces Makefiles, you need to understand a little about *make*. You should also understand something about *cpp*, because *imake* uses *cpp* to do most of its work.

Appendix H, *A Little History*, describes how *imake* came into being.

Appendix I, *Related Documents*, lists some other useful sources of information.

Conventions Used in This Book

The following conventions are used in this book:

Bold	is used to emphasize new terms and concepts when they are introduced.
Italic	is used for options to commands, for file and directory names when they appear in the body of a paragraph, and for program and command names.
`Constant Width`	is used in examples to show the contents of files or the output from commands; and to indicate environment variables, rules, entries, and targets.
`Constant Bold`	is used in examples to show commands or other text that should be typed literally by the user. For example, **`rm foo`** means to type "rm foo" exactly as it appears in the text or example.
`Constant Italic`	is used in code fragments and examples to show variables for which a context-specific substitution should be made. The variable *`filename`*, for example, would be replaced by some actual filename.

Comments and Corrections

If you have comments on this book or suggestions for improvement, or if you find errors, please feel free to contact me by sending electronic mail to *dubois@primate.wisc.edu*.

Acknowledgments

Many people contributed in various ways to this book; those listed here were especially helpful.

— Copyright © 1993 O'Reilly & Associates. All rights reserved. —

Thanks are due to Todd Brunhoff, Jim Fulton, Bob Scheifler, and Stephen Gildea for their willingness to answer my myriad questions about *imake* and the X11 configuration files.

Mary Kay Sherer waded through early drafts and refused to be nice to them. But faithful are the wounds of a friend; her criticisms helped weed out many incomprehensiblenesses.

The staff of O'Reilly & Associates was a pleasure to work with. Adrian Nye provided editorial guidance and oversight and is really the one to whom *Software Portability with imake* owes its existence (the book was his idea; I just wrote it). Laura Parker Roerden copyedited what must have seemed an intractable thicket of words and managed final production with the indispensable help of Clairemarie Fisher O'Leary. Lenny Muellner fielded a constant stream of *troff* questions. Ellie Cutler wrote the index. Jennifer Niederst provided design support. Jeff Robbins produced the figures. And Edie Freedman designed the book and the cover with that marvelous snake. David Brooks, Gary Keim, David Lewis, Andy Oram, and Tom Sauer provided helpful review comments.

Most of all I'd like to thank my wife Karen, who endured author's-widow tribulations with considerable grace and understanding. And patience: she listened while I rehearsed often and at length my thoughts about the topics in this book, even though she "cared for none of those things." This was invaluable in helping me sort out what I was trying to write. Her contribution was significant, and greatly appreciated.

— Copyright © 1993 O'Reilly & Associates. All rights reserved. —