Preface

The hungry sheep look up, and are not fed…
—John Milton, *Lycidas*

This handbook 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 a program 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 (Version 11), Motif, and the Khoros software development environment. X11 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 X11 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 X11 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* documentation has been sparse. 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 handbook 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 handbook will teach you. If you already use *imake*, you'll learn how to use it more effectively. The handbook 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.
- 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 X11 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 X11 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 X11 *contrib* distribution.

- You admire the portability of X11 and want to achieve the same for your own software, but you've found it difficult to use *imake* without a copy of X11 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 handbook 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 consists of appendices containing reference material or covering special topics. 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. In this chapter I assume you want to understand *imake*'s workings in more detail—not just what it does, but how and why.

Chapter 4, *Writing Comments*, describes how to write comments in configuration files and

Imakefiles, and how to deal with problems that arise in connection with commenting. The issue here, as in Chapter 3, is how to write down information about project configuration, so that it is useful to people rather than to programs.

Chapter 5, *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 best-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 6, *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 7, *Imakefile Troubleshooting*, discusses things that can go wrong when you write Imakefiles and how to fix them.

Part II

Chapter 8, *A Closer Look at Makefile Generation*, examines the process by which *imake* builds Makefiles, including a discussion of how the `Makefile` and `Makefiles` target entries work.

Chapter 9, *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 10, *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. This chapter traces the design of *imboot*, a general-purpose *Makefile* bootstrapper, and shows how to use it with various sets of configuration files, such as those from X11, Motif, and OpenWindows. The chapter also develops an alternative approach to generating the `Makefile` target entry.

Chapter 11, *Introduction to Configuration File Writing*, shows by example how to write a set of configuration files. It provides a general description of the various ways you can specify the contents of configuration files.

Chapter 12, *Writing Rule Macros*, continues the discussion begun in Chapter 11, focusing on the design and implementation of *imake* rules.

Chapter 13, *Configuration Problems and Solutions*, discusses several configuration problems and shows how to solve them using the principles and techniques described in Chapters 11 and 12.

Chapter 14, *Troubleshooting Configuration Files*, discusses things that can go wrong when you write configuration files and how to fix them. It complements Chapter 7, *Imakefile Troubleshooting*.

Chapter 15, *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 16, *Creating Extensible Configuration Files*, describes a procedure you can use to convert existing project-specific configuration files to the extensible architecture.

Chapter 17, *Using Extensible Configuration Files*, shows how to develop new projects that take advantage of the flexibility afforded by the extensible architecture.

Chapter 18, *Using imake on Non-UNIX Systems*, covers some of the issues you must address if you're trying to port *imake* to a non-UNIX system. It also discusses writing Imakefiles and configuration files to be less UNIXcentric and so is useful even if your own interest is primarily in UNIX but you wish to make it easier for your projects to be ported to non-UNIX systems by others.

Part III

Appendix A, *Obtaining Configuration Software*, describes how to get the software described in this handbook. *imake* isn't always included as part of the software distributed with the UNIX operating system, but anyone with World Wide Web or FTP access on the Internet can get it. *imake* is also available by electronic mail.

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

Appendix C, *Configuration Programs: A Quick Reference*, documents the software described in this handbook, using an 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, *Other Sources of Information*, lists some other references on *imake*, *make*, and *cpp*. It also provides instructions for getting on the *imake* mailing list.

Appendix J, *Using imake with OpenWindows*, describes some special problems with using *imake* under OpenWindows, and how to handle them.

imake and the X Window System

imake is part of the distribution of the X Window System, Version 11, a product of X Consortium, Inc. X11 (and hence *imake*) is owned and copyrighted by X Consortium, Inc., but is freely available. For more information, the X Consortium is reachable on the World Wide Web at:

```
http://www.x.org
ftp://ftp.x.org
```

Or by surface mail at:

```
X Consortium, Inc.
201 Broadway
Cambridge, MA 02139-1955
USA
```

You don't need to have or use X11 to use *imake*. Nevertheless, X11 has a strong presence in the *imake* world. In particular, the X11 configuration files are very popular, and I refer to them often. When the first edition of this handbook was published, the current release of the X Window System was Version 11, Release 5 (denoted as X11R5, or just R5). For the second edition, Release 6 (X11R6) was current during most of the revision period and Release 6.1 (X11R6.1) was issued shortly before publication.

R6.1 is used as the reference release for most of the discussion in this handbook, but R5 and R6 are mentioned on occasion as well. The default pathnames for program and configuration file installation directories are shown for each release in Table 1. You can use these pathnames to recognize on sight which release particular examples refer to.

Table 0−1: Default X11 Program and Configuration File Installation Directories

X11 Release	Program Directory	Configuration File Directory
X11R5	*/usr/bin/X11*	*/usr/lib/X11/config*
X11R6	*/usr/X11R6/bin*	*/usr/X11R6/lib/X11/config*
X11R6.1	*/usr/X11R6.1/bin*	*/usr/X11R6.1/lib/X11/config*

Conventions Used in This Handbook

The following typographical conventions are used in this handbook:

Italic
is used for file and directory names when they appear in the body of a paragraph, for program and command names, and for options to commands.

`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 myfile`** means to type “rm myfile” 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.

Acknowledgments

Many people contributed in various ways to both editions of this handbook; those listed here were especially helpful.

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

David Brooks, Steve Dennis, Gary Keim, Jim Kohli, Steve Kroeker, David Lewis, Miles O’Neal, Andy Oram, and Tom Sauer provided helpful review comments.

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).

For the first edition, 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. For the reprint of this handbook, Nicole Gipson entered new edits, Chris Reilley designed the figure for the new appendix, and Chris Tong prepared the index.

For the second edition, Gigi Estabrook pulled the various pieces together, and Clairemarie Fisher O'Leary coordinated the production. David Sewell copyedited the manuscript, and Nancy Kotary and Evan Garcia made the edits. Seth Maislin helped to update the index. Lenny Muellner fielded yet more *troff* questions, and Chris Reilley did the figures.

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.

We'd Like To Hear from You

We have tested and verified all of the information in this handbook 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.
101 Morris Street
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:

```
info@ora.com
```

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

`bookquestions@ora.com`

To correspond directly with the author, send email to:

`dubois@primate.wisc.edu`