# Preface

Solaris™ provides an environment in which application developers can build applications and libraries using the link-editor `ld(1)`, and execute these utilities with the aid of the runtime linker `ld.so.1`. For many application developers, the fact that the link-editor is called via the compilation system, and that the runtime linker may play a part in the execution of their application, is mildly interesting. This manual is for those who wish to understand more fully the concepts involved.

## About This Manual

This manual describes the operations of the Solaris link-editor and runtime linker. Special emphasis is placed on the generation and use of shared objects because of their importance in a dynamic runtime environment.

### Intended Audience

This manual is intended for a range of programmers who are interested in the Solaris linkers, from the curious beginner to the advanced user:

- Beginners learn the principle operations of the link-editor and runtime linker.
- Intermediate programmers learn to build, and use, efficient custom libraries.
- Advanced programmers, such as language-tools developers, learn how to interpret and generate object files.

Not many programmers should find it necessary to read this manual from cover to cover.

## *Organization*

Chapter 1, “Introduction”, gives an overview of the linking processes under Solaris. This chapter is intended for all programmers.

Chapter 2, “Link-Editor”, describes the functions of the link-editor, its two modes of linking (*static* and *dynamic*), scope and forms of input, and forms of output. This chapter is intended for all programmers.

Chapter 3, “Runtime Linker”, describes the execution environment and program-controlled runtime binding of code and data. This chapter is intended for all programmers.

Chapter 4, “Shared Objects”, gives definitions of shared objects, describes their mechanisms, and explains how to build and use them. This chapter is intended for all programmers.

Chapter 5, “Versioning”, describes how to manage the evolution of an interface provided by a dynamic object.

Chapter 6, “Object Files”, is a reference chapter on ELF files. This chapter is intended for advanced programmers.

Chapter 7, “Mapfile Option”, describes the mapfile directives to the linker, which specify the layout of the output file. This chapter is intended for advanced programmers.

Appendix A, “Link-Editor Quick Reference”, gives an overview of the most commonly used link-editor options, and is intended for all programmers.

Appendix B, “Versioning Quick Reference”, gives naming conventions and guidelines for versioning shared objects, and is intended for all programmers.

Throughout this document, all command-line examples use `sh(1)` syntax, and all programming examples are written in the C language.