Advanced Linux Programming

Contents At a Glance

Advanced Linux Programming

Mark Mitchell, Jeffrey Oldham,
and Alex Samuel

www.newriders.com
201 West 103rd Street, Indianapolis, Indiana 46290
An Imprint of Pearson Education
Boston • Indianapolis • London • Munich • New York • San Francisco

Advanced Linux Programming

FIRST EDITION: June, 2001

International Standard Book Number: 0-7357-1043-0

Library of Congress Catalog Card Number: *00-105343*

05 04 03 02 01 7 6 5 4 3 2 1

Interpretation of the printing code: The rightmost double-digit number is the year of the book's printing; the rightmost single-digit number is the number of the book's printing. For example, the printing code 01-1 shows that the first printing of the book occurred in 2001.

Composed in Bembo and MCPdigital by New Riders Publishing.

Printed in the United States of America.

Trademarks

Warning and Disclaimer

Publisher
David Dwyer

Associate Publisher
Al Valvano

Executive Editor
Stephanie Wall

Managing Editor
Gina Brown

Acquisitions Editor
Ann Quinn

Development Editor
Laura Loveall

Product Marketing Manager
Stephanie Layton

Publicity Manager
Susan Petro

Project Editor
Caroline Wise

Copy Editor
Krista Hansing

Senior Indexer
Cheryl Lenser

Manufacturing Coordinator
Jim Conway

Book Designer
Louisa Klucznik

Cover Designer
Brainstorm Design, Inc.

Cover Production
Aren Howell

Proofreader
Debra Neel

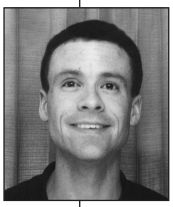
Composition
Amy Parker

Table of Contents

Table of Program Listings

About the Authors

Mark Mitchell received a bachelor of arts degree in computer science from Harvard in 1994 and a master of science degree from Stanford in 1999. His research interests centered on computational complexity and computer security. Mark has participated substantially in the development of the GNU Compiler Collection, and he has a strong interest in developing quality software.



Jeffrey Oldham received a bachelor of arts degree in computer science from Rice University in 1991. After working at the Center for Research on Parallel Computation, he obtained a doctor of philosophy degree from Stanford in 2000. His research interests center on algorithm engineering, concentrating on flow and other combinatorial algorithms. He works on GCC and scientific computing software.

Alex Samuel graduated from Harvard in 1995 with a degree in physics. He worked as a software engineer at BBN before returning to study physics at Caltech and the Stanford Linear Accelerator Center. Alex administers the Software Carpentry project and works on various other projects, such as optimizations in GCC.

Mark and Alex founded **CodeSourcery LLC** together in 1999. Jeffrey joined the company in 2000. CodeSourcery's mission is to provide development tools for GNU/Linux and other operating systems; to make the GNU tool chain a commercial-quality, standards-conforming development tool set; and to provide general consulting and engineering services. CodeSourcery's Web site is `http://www.codesourcery.com`.

About the Technical Reviewers

These reviewers contributed their considerable hands-on expertise to the entire development process for *Advanced Linux Programming*. As the book was being written, these dedicated professionals reviewed all the material for technical content, organization, and flow. Their feedback was critical to ensuring that *Advanced Linux Programming* fits our reader's need for the highest quality technical information.

Glenn Becker has many degrees, all in theatre. He presently works as an online producer for SCIFI.COM, the online component of the SCI FI channel, in New York City. At home he runs Debian GNU/Linux and obsesses about such topics as system administration, security, software internationalization, and XML.

John Dean received a BSc(Hons) from the University of Sheffield in 1974, in pure science. As an undergraduate at Sheffield, John developed his interest in computing. In 1986 he received a MSc from Cranfield Institute of Science and Technology in Control Engineering. While working for Roll Royce and Associates, John became involved in developing control software for computer-aided inspection equipment of nuclear steam-raising plants. Since leaving RR&A in 1978, he has worked in the petrochemical industry developing and maintaining process control software. John worked a volunteer software developer for MySQL from 1996 until May 2000, when he joined MySQL as a full-time employee. John's area of responsibility is MySQL on MS Windows and developing a new MySQL GUI client using Trolltech's Qt GUI application toolkit on both Windows and platforms that run X-11.

Acknowledgments

We greatly appreciate the pioneering work of Richard Stallman, without whom there would never have been the GNU Project, and of Linus Torvalds, without whom there would never have been the Linux kernel. Countless others have worked on parts of the GNU/Linux operating system, and we thank them all.

We thank the faculties of Harvard and Rice for our undergraduate educations, and Caltech and Stanford for our graduate training. Without all who taught us, we would never have dared to teach others!

W. Richard Stevens wrote three excellent books on UNIX programming, and we have consulted them extensively. Roland McGrath, Ulrich Drepper, and many others wrote the GNU C library and its outstanding documentation.

Robert Brazile and Sam Kendall reviewed early outlines of this book and made wonderful suggestions about tone and content. Our technical editors and reviewers (especially Glenn Becker and John Dean) pointed out errors, made suggestions, and provided continuous encouragement. Of course, any errors that remain are no fault of theirs!

Thanks to Ann Quinn, of New Riders, for handling all the details involved in publishing a book; Laura Loveall, also of New Riders, for not letting us fall too far behind on our deadlines; and Stephanie Wall, also of New Riders, for encouraging us to write this book in the first place!

Tell Us What You Think

As the reader of this book, you are the most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

As the Executive Editor for the Web Development team at New Riders Publishing, I welcome your comments. You can fax, email, or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books stronger.

Please note that I cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail I receive, I might not be able to reply to every message.

When you write, please be sure to include this book's title and author, as well as your name and phone or fax number. I will carefully review your comments and share them with the author and editors who worked on the book.

Fax:	317-581-4663
Email:	`Stephanie.Wall@newriders.com`
Mail:	Stephanie Wall Executive Editor New Riders Publishing 201 West 103rd Street Indianapolis, IN 46290 USA

Introduction

GNU/Linux has taken the world of computers by storm. At one time, personal computer users were forced to choose among proprietary operating environments and applications. Users had no way of fixing or improving these programs, could not look "under the hood," and were often forced to accept restrictive licenses. GNU/Linux and other open source systems have changed that—now PC users, administrators, and developers can choose a free operating environment complete with tools, applications, and full source code.

A great deal of the success of GNU/Linux is owed to its open source nature. Because the source code for programs is publicly available, everyone can take part in development, whether by fixing a small bug or by developing and distributing a complete major application. This opportunity has enticed thousands of capable developers worldwide to contribute new components and improvements to GNU/Linux, to the point that modern GNU/Linux systems rival the features of any proprietary system, and distributions include thousands of programs and applications spanning many CD-ROMs or DVDs.

The success of GNU/Linux has also validated much of the UNIX philosophy. Many of the application programming interfaces (APIs) introduced in AT&T and BSD UNIX variants survive in Linux and form the foundation on which programs are built. The UNIX philosophy of many small command line-oriented programs working together is the organizational principle that makes GNU/Linux so powerful. Even when these programs are wrapped in easy-to-use graphical user interfaces, the underlying commands are still available for power users and automated scripts.

A powerful GNU/Linux application harnesses the power of these APIs and commands in its inner workings. GNU/Linux's APIs provide access to sophisticated features such as interprocess communication, multithreading, and high-performance networking. And many problems can be solved simply by assembling existing commands and programs using simple scripts.

GNU and Linux

Where did the name GNU/Liux come from? You've certainly heard of Linux before, and you may have heard of the GNU Project. You may not have heard the name GNU/Linux, although you're probably familiar with the system it refers to.

Linux is named after Linus Torvalds, the creator and original author of the *kernel* that runs a GNU/Linux system. The kernel is the program that performs the most basic functions of an operating system: It controls and interfaces with the computer's hardware, handles allocation of memory and other resources, allows multiple programs to run at the same time, manages the file system, and so on.

The kernel by itself doesn't provide features that are useful to users. It can't even provide a simple prompt for users to enter basic commands. It provides no way for users to manage or edit files, communicate with other computers, or write other programs. These tasks require the use of a wide array of other programs, including command shells, file utilities, editors, and compilers. Many of these programs, in turn, use libraries of general-purpose functions, such as the library containing standard C library functions, which are not included in the kernel.

On GNU/Linux systems, many of these other programs and libraries are software developed as part of the GNU Project.[1] A great deal of this software predates the Linux kernel. The aim of the GNU Project is "to develop a complete UNIX-like operating system which is free software" (from the GNU Project Web site, `http://www.gnu.org`).

The Linux kernel and software from the GNU Project has proven to be a powerful combination. Although the combination is often called "Linux" for short, the complete system couldn't work without GNU software, any more than it could operate without the kernel. For this reason, throughout this book we'll refer to the complete system as GNU/Linux, except when we are specifically talking about the Linux kernel.

The GNU General Public License

The source code contained in this book is covered by the GNU *General Public License* (*GPL*), which is listed in Appendix F, "GNU General Public License." A great deal of free software, especially GNU/Linux software, is licensed under it. For instance, the Linux kernel itself is licensed under the GPL, as are many other GNU programs and libraries you'll find in GNU/Linux distributions. If you use the source code in this book, be sure to read and understand the terms of the GPL.

The GNU Project Web site includes an extensive discussion of the GPL (`http://www.gnu.org/copyleft/`) and other free software licenses. You can find information about open source software licenses at `http://www.opensource.org/licenses/index.html`.

Who Should Read This Book?

This book is intended for three types of readers:

- You might be a developer already experienced with programming for the GNU/Linux system, and you want to learn about some of its advanced features and capabilities. You might be interested in writing more sophisticated programs with features such as multiprocessing, multithreading, interprocess communication, and interaction with hardware devices. You might want to improve your programs by making them run faster, more reliably, and more securely, or by designing them to interact better with the rest of the GNU/Linux system.

1. GNU is a recursive acronym: It stands for "GNU's Not UNIX."

- You might be a developer experienced with another UNIX-like system who's interested in developing GNU/Linux software, too. You might already be familiar with standard APIs such as those in the POSIX specification. To develop GNU/Linux software, you need to know the peculiarities of the system, its limitations, additional capabilities, and conventions.
- You might be a developer making the transition from a non-UNIX environment, such as Microsoft's Win32 platform. You might already be familiar with the general principles of writing good software, but you need to know the specific techniques that GNU/Linux programs use to interact with the system and with each other. And you want to make sure your programs fit naturally into the GNU/Linux system and behave as users expect them to.

This book is not intended to be a comprehensive guide or reference to all aspects of GNU/Linux programming. Instead, we'll take a tutorial approach, introducing the most important concepts and techniques, and giving examples of how to use them. Section 1.5, "Finding More Information," in Chapter 1, "Getting Started," contains references to additional documentation, where you can obtain complete details about these and other aspects of GNU/Linux programming.

Because this is a book about advanced topics, we'll assume that you are already familiar with the C programming language and that you know how to use the standard C library functions in your programs. The C language is the most widely used language for developing GNU/Linux software; most of the commands and libraries that we discuss in this book, and most of the Linux kernel itself, are written in C.

The information in this book is equally applicable to C++ programs because that language is roughly a superset of C. Even if you program in another language, you'll find this information useful because C language APIs and conventions are the *lingua franca* of GNU/Linux.

If you've programmed on another UNIX-like system platform before, chances are good that you already know your way around Linux's low-level I/O functions (`open`, `read`, `stat`, and so on). These are different from the standard C library's I/O functions (`fopen`, `fprintf`, `fscanf`, and so on). Both are useful in GNU/Linux programming, and we use both sets of I/O functions throughout this book. If you're not familiar with the low-level I/O functions, jump to the end of the book and read Appendix B, "Low-Level I/O," before you start Chapter 2, "Writing Good GNU/Linux Software."

This book does not provide a general introduction to GNU/Linux systems. We assume that you already have a basic knowledge of how to interact with a GNU/Linux system and perform basic operations in graphical and command-line environments. If you're new to GNU/Linux, start with one of the many excellent introductory books, such as Michael Tolber's *Inside Linux* (New Riders Publishing, 2001).

Conventions

This book follows a few typographical conventions:

- A new term is set in *italics* the first time it is introduced.
- Program text, functions, variables, and other "computer language" are set in a fixed-pitch font—for example, `printf ("Hello, world!\bksl n")`.
- Names of commands, files, and directories are also set in a fixed-pitch font—for example, `cd /`.
- When we show interactions with a command shell, we use `%` as the shell prompt (your shell is probably configured to use a different prompt). Everything after the prompt is what you type, while other lines of text are the system's response.

  For example, in this interaction

  ```
  % uname
  Linux
  ```

  the system prompted you with `%`. You entered the `uname` command. The system responded by printing `Linux`.
- The title of each source code listing includes a filename in parentheses. If you type in the listing, save it to a file by this name. You can also download the source code listings from the *Advanced Linux Programming* Web site (`http://www.newriders.com` or `http://www.advancedlinuxprogramming.com`).

We wrote this book and developed the programs listed in it using the Red Hat 6.2 distribution of GNU/Linux. This distribution incorporates release 2.2.14 of the Linux kernel, release 2.1.3 of the GNU C library, and the EGCS 1.1.2 release of the GNU C compiler. The information and programs in this book should generally be applicable to other versions and distributions of GNU/Linux as well, including 2.4 releases of the Linux kernel and 2.2 releases of the GNU C library.