Embedded Linux System Design and Development

P. Raghavan • Amol Lad • Sriram Neelakandan

Auerbach Publications
Taylor & Francis Group

EMBEDDED LINUX SYSTEM DESIGN AND DEVELOPMENT

P. Raghavan • Amol Lad • Sriram Neelakandan

Auerbach Publications
Taylor & Francis Group
Boca Raton New York

Published in 2006 by
Auerbach Publications
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3

International Standard Book Number-10: 0-8493-4058-6 (Hardcover)
International Standard Book Number-13: 978-0-8493-4058-1 (Hardcover)
Library of Congress Card Number 2005048179

Library of Congress Cataloging-in-Publication Data

Raghavan, P. (Pichai), 1973-
Embedded Linux system design and development / P. Raghavan, Amol Lad, Sriram Neelakandan.
p. cm.
Includes bibliographical references and index.
ISBN 0-8493-4058-6 (alk. paper)
1. Linux. 2. Operating systems (Computers) 3. Embedded computer systems. I. Lad, Amol. II. Neelakandan, Sriram. III. Title.

QA76.76.O63R335 2005
005.4'32--dc22 2005048179

T&F informa

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the Auerbach Publications Web site at
http://www.auerbach-publications.com

EMBEDDED LINUX SYSTEM DESIGN AND DEVELOPMENT

Other Auerbach Publications in Software Development, Software Engineering, and Project Management

The Complete Project Management Office Handbook
Gerard M. Hill
0-8493-2173-5

Complex IT Project Management: 16 Steps to Success
Peter Schulte
0-8493-1932-3

Creating Components: Object Oriented, Concurrent, and Distributed Computing in Java
Charles W. Kann
0-8493-1499-2

The Hands-On Project Office: Guaranteeing ROI and On-Time Delivery
Richard M. Kesner
0-8493-1991-9

Interpreting the CMMI®: A Process Improvement Approach
Margaret Kulpa and Kent Johnson
0-8493-1654-5

ISO 9001:2000 for Software and Systems Providers: An Engineering Approach
Robert Bamford and William John Deibler II
0-8493-2063-1

The Laws of Software Process: A New Model for the Production and Management of Software
Phillip G. Armour
0-8493-1489-5

Real Process Improvement Using the CMMI®
Michael West
0-8493-2109-3

Six Sigma Software Development
Christine Tayntor
0-8493-1193-4

Software Architecture Design Patterns in Java
Partha Kuchana
0-8493-2142-5

Software Configuration Management
Jessica Keyes
0-8493-1976-5

Software Engineering for Image Processing
Phillip A. Laplante
0-8493-1376-7

Software Engineering Handbook
Jessica Keyes
0-8493-1479-8

Software Engineering Measurement
John C. Munson
0-8493-1503-4

Software Metrics: A Guide to Planning, Analysis, and Application
C.R. Pandian
0-8493-1661-8

Software Testing: A Craftsman's Approach, Second Edition
Paul C. Jorgensen
0-8493-0809-7

Software Testing and Continuous Quality Improvement, Second Edition
William E. Lewis
0-8493-2524-2

IS Management Handbook, 8th Edition
Carol V. Brown and Heikki Topi, Editors
0-8493-1595-9

Lightweight Enterprise Architectures
Fenix Theuerkorn
0-8493-2114-X

Outsourcing Software Development Offshore: Making It Work
Tandy Gold
0-8493-1943-9

Maximizing ROI on Software Development
Vijay Sikka
0-8493-2312-6

Implementing the IT Balanced Scorecard
Jessica Keyes
0-8493-2621-4

AUERBACH PUBLICATIONS

www.auerbach-publications.com
To Order Call: 1-800-272-7737 • Fax: 1-800-374-3401
E-mail: orders@crcpress.com

All source code in the book is released under GNU GPL v2. It can be used as desired under terms and conditions of GNU GPL v2.

Trademarks

- MIPS is a registered trademark and YAMON is a trademark of MIPS Technologies.
- IBM and ClearCase are registered trademarks and PowerPC is a trademark of International Business Machines Corporation.
- UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.
- X11 is a trademark of Massachusetts Institute of Technology.
- NEC is a registered trademark of NEC Corporation.
- HP is a registered trademark of Hewlett-Packard Company.
- ColdFire is a registered trademark and Motorola is a trademark of Motorola, Inc.
- Microblaze is trademark of Xilinx Inc.
- Red Hat is a registered trademark and eCos and RedBoot are trademarks of Red Hat, Inc.
- uClinux is a registered trademark of Arcturus Networks Inc.
- Linux is a registered trademark of Linus Torvalds.
- GoAhead is a registered trademark of GoAhead Software, Inc.
- RTLinux is a registered trademark and FSMLabs, RTLinuxPro and RTCore are trademarks of Finite State Machine Labs, Inc.
- Debian is a registered trademark of Software in the Public Interest, Inc.
- LMBench is a trademark of BitMover, Inc.
- VRTX is a trademark of Microtech Research Inc.
- VxWorks and pSOS are registered trademarks of Wind River Systems, Inc.
- Trolltech is a registered trademark and Qt is a trademark of Trolltech in Norway, the United States and other countries.
- OpenGL is a registered trademark of Silicon Graphics, Inc.
- Perforce is a registered trademark of Perforce Software, Inc.
- Eclipse is a trademark of Eclipse Foundation, Inc.
- KDE and K Desktop Environment are trademarks of KDE.
- FFmpeg is a trademark of Fabrice Bellard, originator of the FFmpeg project.
- NVIDIA is a registered trademark of NVIDIA Corporation in the United States and other countries.
- ViewML is a registered trademark of Century Software Inc.
- QNX and Neutrino are registered trademarks of QNX Software Systems Ltd.
- Nucleus is a trademark of Accelerated Technology, Inc.
- Accelerated Technology is a registered trademark of Mentor Graphics Corporation.
- ARM and StrongARM are registered trademarks and ARM7 and ARM9 are trademarks of Advanced RISC Machines, Ltd.
- AMD is a registered trademark of Advanced Micro Devices, Inc.
- Intel and Pentium are registered trademarks and i386 and XScale are trademarks of Intel Corporation.
- Sharp is a registered trademark of Sharp Electronics Corp.
- SPARC is a registered trademark of SPARC International, Inc., and is used under license by Sun Microsystems, Inc.
- Toshiba is a registered trademark of the Toshiba Corporation.
- MontaVista is registered trademark of MontaVista Software Inc.
- LynxOS and BlueCat are registered trademarks and LynuxWorks, SpyKer and VisualLynux are trademarks of LynuxWorks, Inc.
- Samsung is a registered trademark of Samsung Electronics America, Inc. and its related entities.
- Ericsson is a registered trademark of Ericsson, Inc.
- Atmel is registered trademarks of Atmel Corporation.
- TimeSys®, TimeStorm®, TimeStorm IDE™, TimeStorm LVS™, TimeStorm LDS™, TimeStorm LHD™, TimeSys Reservations™, TimeTrace®, Linux/RT™ and TimeWiz® are registered or unregistered trademarks of TimeSys Corporation in the United States and other countries.
- NeoMagic is a registered trademark of NeoMagic Corporation.
- Transmeta is a trademark of Transmeta Corporation.
- Broadcom is a registered trademark of Broadcom Corporation and/or its subsidiaries.
- SuSE is a registered trademark of SuSE AG.

Dedication

Raghavan

In memory of my late father

Amol

To Lord Kṛṣṇa, my parents, my wife Parul, and my brother Amit

Sriram

To my family and all Linux enthusiasts

Foreword

The industrial revolution appears as a knife-edge change from a rural self-employed lifestyle to a clock-punching, whistle-blowing corporate urban way of life. Being in the middle of the current revolution makes it hard to realize that in fifty years most people will consider the messy, dynamic, no-rules embedded product development environment of today as an obvious clean transition caused by technological changes.

The first embedded software project I worked on didn't use an off-the-shelf operating system—there was none. It wasn't until several years later that WindRiver introduced VxWorks®. In the mid-1990s it appeared that nothing could unseat VxWorks; yet, recently WindRiver announced a Linux-based product. Why the change? Today the most common embedded operating system used in new products is Linux.

For fourteen years I was part of a small army of firmware engineers working on the development of HP LaserJet™ printers. The printer used a homegrown operating system that as I recall was called LaserJet O.S. Usually the very best engineers worked on supporting and extending the operating system. Any LaserJet O.S. documentation that existed, engineers had created. Any test suite was similarly a burden placed on the engineer's shoulders. The effort and expense of these highly talented engineers seldom led to any features that differentiated the product from the competitors. The most important lesson I learned from the experience was to always put your most talented engineers on the features that make your product unique and outsource the infrastructure. Embedded Linux is often the best choice for the operating system infrastructure for products needing nontrivial connectivity.

Whether you support Linux in-house or purchase a Linux board support package for your processor, you will still need to understand the overall system and at times the details of a particular subsystem. In this book the authors have done a good job fitting all the pieces together that are necessary for embedded Linux development. The book discusses topics such as board support packages, embedded storage, and real-time Linux programming in

depth. Embedded graphics and uClinux are also explained with clarity. The book is a good attempt to address the concerns of an embedded Linux developer.

The rapid growth of Linux as the top choice for an embedded operating system in new products is in part due to the ease of using embedded Linux to replace homegrown operating systems. Although this book is specifically for running Linux on embedded systems it can also be used as a guide to port a system from a traditional RTOS or homegrown operating system to embedded Linux. It may be the need for TCP/IP networking, USB support, SecureDigital support, or some other standard that causes a company to dump their current operating system and switch to Linux. But it is the joy of developing with Linux that keeps the engineers promoting it for future products.

An astounding amount of Linux information is available on the Web. I suspect it is the most extensively documented software ever. How can a book about embedded Linux provide value over what is already available? First, the scope of embedded Linux and related applications is so large that getting a feel for what is available and what can be done is challenging. Seeing all the pieces separately and working together can help you make sense of the embedded Linux ecosystem. Second, there are technical reasons for needing the right information. In an embedded device, the bootloader, kernel, and file system containing the applications all need to be developed in concert for the solution to work properly. Understanding the interdependencies and getting the development environment to properly build all three images is not straightforward. Also, when you encounter a problem, understanding the tools available to debug the problem and knowing the techniques used for debugging embedded devices can save a significant amount of time and effort.

Finally, the best reason for reading this book on embedded Linux is because the technology is so fascinating. Anyone who had developed embedded products the old way, with one single executable image, will be amazed at the flexibility and power of using embedded Linux. Anyone new to embedded development will find most of the power and flexibility available on their desktop PC works the same in their embedded development environment.

Todd Fischer
President and Founder
Cadenux

Preface

When we were in college in the mid-1990s we heard of an exciting new technology called the Internet that was to have a profound impact on our lives. Along with the Internet we also heard of an open source operating system, Linux, which was being developed by hundreds of programmers around the world. Linux gave us an opportunity to understand the internals of the operating system and we quickly became Linux enthusiasts. We realized that Linux was more than an operating system; here was a movement with few parallels in human history as it was based on the concepts of human dignity, choice, and freedom. Linux gave young programmers like us the reach to the latest technology.

When we became embedded professionals Linux had yet to make a strong presence in the embedded market. However, we were hearing of some exciting improvements such as running a hard real-time kernel along with the Linux kernel and running Linux on MMU-less microcontrollers. Our happiness grew unbounded when we were asked by a customer to move our software on a MIPS-based SoC from a commercial RTOS to embedded Linux. Our experience revealed that the road to embedded Linux is not a very smooth ride. Some of the main reasons were:

1. There is undoubtedly lots of information about embedded Linux on the Internet but it is too scattered to give a consolidated view. Converting this information into a knowledge base can be a time-consuming task. Most of the product-based companies are normally short on time. Decisions need to be made quickly and executed quickly. However, a wrong decision especially on crucial issues such as licensing can prove disastrous to the company.
2. There is a gross misconception that embedded systems are all about the hardware or the operating system. As computing power increases rapidly as per Moore's law the amount of application software that goes into the embedded system has also increased at the same rate. Hence the applications have become the USP for the embedded system. So building a

Linux-based embedded system does not stop with the OS but has to do a lot with writing and building applications. And applications have their own set of issues that are different from the operating system such as licensing, toolchains, and so on.

3. Unlike a commercial RTOS, which gives a single point of support such as patches and documentation, embedded Linux takes a whole new development paradigm. Often the developers need to search for patches or for new information from the various mailing lists. And this can be very time consuming.

When we came out successfully with an embedded Linux design with a variety of applications, we decided to share some of our thoughts and experiences with the rest of the world. The result of that thought process is this book. This book contains an entire development roadmap for embedded Linux systems. Our primary aim is to make the reader aware of the various issues that arise out of embedded Linux development.

The theme of the book is twofold:

- To facilitate movement to embedded Linux from a traditional RTOS
- To explain the system design model with embedded Linux

Benefits to the Reader

The book offers solutions to problems that a developer faces when programming in an embedded Linux environment. Some of the common problems are:

- Understand the embedded Linux development model.
- Write, debug, and profile applications and drivers in embedded Linux.
- Understand embedded Linux BSP architecture.

The book offers practical solutions to the above problems.

After reading this book the reader will

- Understand the embedded Linux development environment.
- Understand and create Linux BSP for a hardware platform.
- Understand the Linux model for embedded storage and write drivers and applications for the same.
- Understand various embedded Linux drivers such as serial, I2C, and so on.
- Port applications to embedded Linux from a traditional RTOS.
- Write real-time applications in embedded Linux.
- Learn methods to find memory leaks and memory corruption in applications and drivers.
- Learn methods to profile applications and the kernel.
- Understand uCLinux architecture and its programming model.
- Understand the embedded Linux graphics subsystem.

The book is also an aid to managers in choosing an embedded Linux distribution, creating a roadmap for the transition to embedded Linux, and applying the Linux licensing model in a commercial product.

Audience

Primary Audience

- *Architects:* They are more concerned with real-time issues, performance, and porting plans.
- *Software programmers:* They need to get into the minute details of the technology.

Secondary Audience

- *Legal staff:* Because most embedded products involve intellectual property, any wrong understanding of the licensing issues can prove detrimental to the company itself.
- *Managers:* They are normally concerned about choosing the distribution, version, toolset, and vendor.
- *Testing and support team:* Because the look and feel of the product can change when moving to embedded Linux, the test and support team needs to be educated.

Background

The authors expect a basic understanding of embedded system programming in any embedded OS from the reader. The book is not a Linux kernel book. Familiarity with basic Linux kernel concepts and the user-space programming model is desirable.

The book attempts to be independent of the kernel version; however, wherever necessary the 2.4 or the 2.6 kernels are used as examples.

Downloading Source Code

Readers can download source code from the following URL: http://www.crcpress.com/e_products/downloads/download.asp?cat_no=AU0586

Acknowledgments

I thank the management of my present employer, Philips, for giving me the support to go ahead with the book. Any work of mine has always been incomplete without the blessings of my dear mother. And last but not least I would like to thank my wife, Bhargavi, for spending some cold days alone when I was busy penning down the pages for this book.

Raghavan

I would like to thank all the people who made this work possible. First my mother, who used to tell me to work for the book whenever she saw me roaming here and there, like any mother telling her kid to study. I also express thanks to my father who kept on asking me about the status of the manuscript, like a project manager. I thank my wife, Parul, for her patience during manuscript preparation. I remember when Raghav told me about this project and asked me to join. It was just two months after my marriage. I thank Parul for her encouragement and also thank her for helping me out in formatting the manuscript.

Amol

Thanks to Raghav who had the idea of writing this book. I still remember the first meeting when he instilled the confidence in me to take up this work. I thank my dad, mom, and sister for their support. Thanks to the entire "boys" gang at Bangalore who have been kind enough to share the powerful "Athlon/Audigy/ATI Radeon 9500" game PC for mean activities such as running Linux and typing sample code. They consider running a word processor on such a PC as a gross waste of computing power.

Sriram

We take this opportunity to thank Todd Fischer, president and founder, Cadenux, for giving us time from his busy schedule to write the foreword for the book. We thank David McCullogh, one of the uClinux core maintainers,

and Dr. Paul Dale for reviewing the chapter on uClinux and for providing their valuable comments. We also thank Greg Haerr, CEO of Century Software and founder of the Nano-X windowing system, for his valuable review comments on the embedded graphics chapter. We thank Satish MM, director of Verismo Networks, for his valuable comments on GPL. We thank our close friend and guide, Deepak Shenoy, for coming up with the idea to write a book based on our development experience. Finally we thank all Linux kernel developers and user-space programmers for taking Linux to new heights.

Introduction

The text is divided into ten chapters and two appendices.

Chapter 1, "Introduction," gives a brief history of embedded Linux and what the benefits of embedded Linux are over other RTOSs. It discusses in detail the features of various open source and commercial embedded Linux distributions available. The chapter concludes by presenting a transition roadmap from a traditional RTOS to embedded Linux.

Chapter 2, "Getting Started," explains the architecture of embedded Linux and compares it with traditional RTOS and microkernel architectures. In brief various Linux kernel subsystems such as the hardware abstraction layer, memory management, scheduler, file system, and so on are given. A small description of the user-space Linux programming model is also given. The second half of the chapter explains the Linux start-up sequence, from bootloaders to kernel start-up and user-space start-up scripts. The last section explains the steps involved in building a GNU cross-platform toolchain.

Chapter 3, "Board Support Package," explains bootloader architecture followed by a discussion on the system memory map, both hardware and software memory maps. The second half of the chapter explains interrupt management, the PCI subsystem, timers, UART, and power management in detail.

Chapter 4, "Embedded Storage," explains the MTD subsystem architecture for accessing flash devices. The second half of the chapter discusses various embedded file systems such as RAMFS, CRAMFS, JFFS2, NFS, and so on. The chapter also discusses various methods for optimizing storage space in an embedded system, both kernel and user-space optimizations. A discussion of various applications designed for embedded Linux such as Busybox is given. Finally some steps for tuning the kernel memory are given.

Chapter 5, "Embedded Drivers," discusses in detail various embedded drivers such as the Serial driver, Ethernet driver, I2C subsystem, and USB gadgets.

Chapter 6, "Porting Applications," discusses an application porting roadmap from a traditional RTOS to embedded Linux. The rest of the chapter explains the porting roadmap in detail. First a discussion on Linux pthreads is given, then the Operating System Porting Layer (OSPL), and finally a kernel API driver.

Chapter 7, "Real-Time Linux," discusses the real-time features in Linux. It explains the various latencies involved in the kernel such as interrupt and scheduling latency and efforts that are made to improve the kernel response time such as kernel preemption and O(1) scheduler. The core of the chapter is the discussion of POSIX.1b programming interfaces in Linux. The chapter explains various POSIX.1b real-time extensions such as real-time schedulers, memory locking, message queues, semaphores, and asynchronous I/O in detail. The last section explains in brief the hard real-time approach to Linux followed by a real-time programming model in RTAI.

Chapter 8, "Building and Debugging," is divided into three sections: building, debugging, and profiling. The first section explains various mechanisms for building kernel and user-space applications. In the second section tools such as mtrace, dmalloc, and valgrind to debug memory problems are explained. Finally the last section discusses eProf, OProfile, and kernel function instrumentation profiling methods to profile user-space and kernel functions.

Chapter 9, "Embedded Graphics," explains in detail a generic frame buffer driver and how to write applications using the frame buffer interface. It also discusses in brief the X graphics subsystem and why it is not suitable for embedded devices. The last section explains the Nano-X windowing environment.

Chapter 10, "uClinux," explains the architecture and programming environment in uClinux. The first half of the chapter explains the bFLT executable file format and how programs are loaded and executed in uClinux-based systems. Next a discussion about memory management, process creation, and shared libraries in uClinux is given. The final section explains XIP and how to port applications from standard Linux to uClinux. It also explains how to build applications for uClinux.

Appendix A, "Booting Faster," explains various techniques to reduce Linux boot-up time.

Appendix B, "GPL and Embedded Linux," discusses what GPL means to embedded Linux and how proprietary software can be kept safe with embedded Linux.

Source code is available for downloading from http://www.crcpress.com/e_products/downloads/download.asp?cat_no=AU0586