
Undergraduate Topics in Computer Science

Undergraduate Topics in Computer Science (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems. Many include fully worked solutions.

More information about this series at <http://www.springer.com/series/7592>

Alan Holt · Chi-Yu Huang

Embedded Operating Systems

A Practical Approach

Alan Holt
IP Performance
Bristol
UK

Chi-Yu Huang
Tata Technologies Ltd.
Bristol
UK

Series editor
Ian Mackie

Advisory Board

Samson Abramsky, University of Oxford, Oxford, UK
Karin Breitman, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil
Chris Hankin, Imperial College London, London, UK
Dexter Kozen, Cornell University, Ithaca, USA
Andrew Pitts, University of Cambridge, Cambridge, UK
Hanne Riis Nielson, Technical University of Denmark, Kongens Lyngby, Denmark
Steven Skiena, Stony Brook University, Stony Brook, USA
Iain Stewart, University of Durham, Durham, UK

ISSN 1863-7310
ISBN 978-1-4471-6602-3
DOI 10.1007/978-1-4471-6603-0

ISSN 2197-1781 (electronic)
ISBN 978-1-4471-6603-0 (eBook)

Library of Congress Control Number: 2014948743

Springer London Heidelberg New York Dordrecht

© Springer-Verlag London 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

*In memory of Marjorie Rose Holt
and Siou-yìn Zheng Huang*

Preface

Most people are aware of mainstream operating systems such as Microsoft's Windows or Mac OS X. This is because these are the operating systems that they directly interact with on their personal computers. However, personal computers rarely operate in isolation nowadays.

Many computing applications require *online* access. Access to the Internet relies on many computing systems, for example, Ethernet switches, packet routers, Wireless access-points and even DNS servers. Unlike personal computers, network infrastructure equipment perform dedicated tasks and are often described as embedded systems.

The term embedded system covers a range of computing systems. It derives from computing systems that are *embedded* within some larger device, for example, within a car's engine management system. The term has clearly broadened to cover standalone systems too. Nor is it limited to network infrastructure systems or consumer goods such as satellite navigation devices.

The term embedded system is applied to devices which are small and have limited resources (relative to a personal computer). Mobile devices such as smart phones and tablets are, therefore, described as embedded devices, even though their function closely resembles a personal computer. Nevertheless, an embedded system needs an operating system specific to its needs, for example:

- Small form factor.
- Low processing power and small memory.
- Reduced power consumption for battery longevity.
- Support for real-time applications.

There are many embedded operating systems available, however, in this book we confine our discussion to the GNU/Linux operating system. GNU/Linux is not exclusively an embedded operating system and is used on personal computers and servers as Windows and OS X. However, due to its open source nature it is highly customisable and can be adapted to many environments. For this reason, GNU/Linux systems have gained considerable popularity in the embedded system market.

This is not to say that GNU/Linux is a panacea for embedded systems and there are many excellent alternatives. Nevertheless, there are some good reasons for choosing GNU/Linux:

- GNU/Linux is open source and is freely available.
- Due to its open source nature, GNU/Linux is highly customisable so we can build bespoke systems specific to our needs.
- It is widely used for embedded systems.
- GNU/Linux, like its Unix predecessor, was used extensively in education to teach operating systems.
- While the subject of the book is embedded operating systems, our choice of GNU/Linux means it could be used as a text for a more general course on operating systems.

Operating systems is a diverse subject area and there many books on the subject and many on GNU/Linux alone, covering topics such as the kernel, administration, networking, wireless, high-performance computing and systems programming (to name a few). There are even books on GNU/Linux embedded systems.

However, we felt there was a gap in the literature which described the component parts of an operating system and how they worked together. We address these issues in this book by adopting a practical approach. The procedure for building each component of the operating system, namely the bootloader, kernel, filesystem, shared libraries, start-up scripts, configuration files and system utilities, from its source code, is described in detail. By the end of this book the reader will be able to build a fully functional GNU/Linux embedded operating system.

We take this opportunity to pre-empt any Amazon reviews stating this book is not for beginners. Indeed, this is not an introductory text on operating systems, rather it is aimed at undergraduate/graduate level students and industry professionals.

Acknowledgments

The authors would like to thank the following people for the valuable contribution they made to this book: Paul While and Edwin Chen.

Contents

1	Introduction	1
1.1	An Overview of Operating Systems	1
1.2	Overview of Embedded Systems	3
1.3	Brief History of Linux and GNU	5
1.4	GNU/Linux as an Embedded Operating System	6
1.5	Virtualisation	7
1.6	Conventions Used in this Book	9
1.7	Book Outline	10
	References	11
2	Overview of GNU/Linux	13
2.1	The Bootloader	14
2.2	The Kernel	15
2.3	The Init Process	16
2.4	The Root Filesystem	17
2.5	Process Management	20
	2.5.1 Signals	20
	2.5.2 Job Control	23
2.6	Process Input/Output	24
2.7	The Process Environment	28
	2.7.1 Shell Parameters	32
	2.7.2 Quoting	33
	2.7.3 Positional Parameters	34
	2.7.4 Special Parameters	35
2.8	Summary	37
	References	37
3	The Filesystem in Detail	39
3.1	GNU/Linux File Space	40
	3.1.1 Permissions	42
3.2	The Filesystem	53
	3.2.1 Pseudo Filesystems	57

3.3	Partitions	59
3.3.1	Partitions and Boot Sectors	59
3.3.2	Extended Partitions.	66
3.4	Summary.	69
	Reference	69
4	Building an Embedded System (First Pass)	71
4.1	Creating the Root Filesystem	72
4.2	Building a Linux Kernel	76
4.3	Build the Root Filesystem	78
4.4	Running UML	79
4.5	Networking	81
4.6	Summary.	84
	Reference	85
5	Building an Embedded System (Second Pass)	87
5.1	Preliminaries	89
5.2	Glibc	89
5.3	Optimisation	92
5.4	Ncurses.	92
5.5	Busybox	93
5.6	Bash.	94
5.7	Sysvinit.	95
5.8	Devices (/dev)	96
5.9	Administrative Files and Directories	97
5.10	Start-up Scripts	102
5.11	Test with UML	104
5.12	Running the System Natively.	105
5.12.1	Compiling the Kernel	106
5.12.2	The Filesystem.	106
5.12.3	The Bootloader	110
5.12.4	Booting the System	111
5.13	Summary.	113
	References	114
6	Compiler Toolchains	115
6.1	GCC.	116
6.1.1	The Preprocessor	117
6.1.2	Optimisation	119
6.1.3	The Assembler.	121
6.1.4	The Linker	122
6.2	Build an ARM Cross Toolchain.	128

6.3	Binutils	130
6.3.1	GCC (Bootstrap Compiler)	131
6.3.2	Newlib	132
6.3.3	GCC.	133
6.4	Testing the Toolchain	134
6.5	Summary.	136
7	Embedded ARM Devices	137
7.1	Raspberry Pi	138
7.1.1	Installing an Operating System.	139
7.1.2	Using the Raspberry Pi Serial Port	140
7.1.3	Remote Serial Port	144
7.2	BeagleBone	151
7.2.1	Setting Up the BeagleBone	152
7.2.2	Physical Computer Programming on the BeagleBone	156
7.3	Summary.	160
	References	160
8	OpenWRT	161
8.1	Open-Mesh	162
8.1.1	Building OpenWRT	163
8.2	Dragino.	167
8.2.1	Booting up the Dragino for the First Time.	168
8.2.2	Running Arduino Yún Firmware	171
8.3	Programming the M32 Unit.	178
8.4	Summary.	181
	References	181
	Appendix A: Start-up Scripts.	183
	Appendix B: Inittab.	191
	Index	193

Abbreviations

API	Application programming interface
Bash	Bourne again shell—a command-line interpreter
Busybox	A collection of Unix utilities designed for embedded systems
Debian	A GNU/Linux distribution
GCC	GNU compiler collection
Glibc	GNU C Library
GNU	GNU is not Unix
Grub	Grand unified bootloader
Ncurses	A library of screen handling functions
NSS	Name switch service
TCP/IP	Transmission control protocol/internet protocol
Tmpfs	Temporary file system
Ubuntu	A GNU/Linux distribution based upon Debian
UML	User mode Linux
Vi	A text editor
VM	Virtual machine