Principles of Operating Systems:
Design & Applications

Brian L. Stuart
FedEx Labs
University of Memphis

Australia · Canada · Mexico · Singapore · Spain · United Kingdom · United States

Principles of Operating Systems:
Design & Applications

Brian L. Stuart

Senior Product Manager:
Alyssa Pratt

Acquisitions Editor:
Amy Jollymore

Development Editor:
Jim Markham

Content Project Manager:
Matt Hutchinson

Marketing Manager:
Bryant Chrzan

Editorial Assistant:
Patrick Frank

Manufacturing Coordinator:
Julio Esperas

Cover Design:
Yvo Riezebos Designs

Art Director:
Beth Paquin

Compositor:
Brian L. Stuart

Printer:
RR Donnelley, Crawfordsville

Printed in the United States of America

For more information, contact Course Technology, 25 Thomson Place,
Boston, Massachusetts, 02210.

Or find us on the World Wide Web at: www.course.com

ISBN 1-4188-3769-5

for my beloved wife and daughter

Brief Contents

Contents

Preface

The seed for this book was planted over 20 years ago when I was in graduate school. During the summer of 1986, a group of students collected for an advanced operating systems seminar, under the leadership of Dr. David Cohen. We began with the intention of writing an operating system, but our focus soon shifted to writing a textbook on operating systems. As so often happens, little of our original intent was accomplished. However, along the way, we had many fruitful discussions about how to organize a book on operating systems and, by extension, how to approach teaching operating systems.

That, combined with a number of years teaching operating systems, led me to observe that there are several different approaches to teaching operating systems. It also led me to realize that no existing text really provided the instructor with the flexibility to draw on each of the different approaches as desired. The motivating objective for this book is to provide that flexibility.

Organization

Seven topics make up the body of this book. I begin with an introduction that highlights history, structure and organization, system calls, and bootstrapping. Following this introduction, I examine in some depth each of the major areas of operating system responsibility: processes, memory, I/O devices, and file systems. The final two topics are security and distributed systems.

The coverage of the first five topics is presented in sequences of four chapters each that examine each topic from a variety of perspectives. The first chapter in each sequence presents general principles associated with managing a resource. In these chapters, I introduce the relevant issues, and I present some standard techniques for addressing those issues. In some cases, the chapters devoted to general principles also include discussion of related issues. For example, the subjects of mutual exclusion and deadlock are discussed along with process management in Chapter 5 because of their relevance to interprocess communication. The second chapter in each sequence surveys a number of historic and current operating systems. The set of nine OS examples includes CTSS, Multics, RT-11, sixth edition UNIX, 4.3BSD, VMS, Windows NT, TinyOS, and Xen. My focus with these is to study in a high-level way how their developers translated available standard techniques into practice. In the third and fourth chapters of each part, I drill down further into implementational considerations. I discuss selected parts of the code for Inferno (in the third chapter) and Linux (in the fourth chapter).

However, I do not use this pattern in Chapters 21 and 22. These chapters discuss security and distributed systems, respectively. Because these topics are extensive enough for full books in their own right, I necessarily take a selective approach to them. These chapters present only a few representative techniques. I also discuss a more restricted set of examples in Chapter 21, illustrating applied security techniques.

I describe how you build a hosted Inferno kernel image in Appendix A. Doing this is a part of implementing solutions to those assignments that ask for modifications to the Inferno kernel. A kernel image built in this way can be run as an application on an existing host OS. Appendix B shows you how to do the same for a native kernel. In particular, I provide the steps necessary to create a bootable floppy image for an x86 PC. This image can then be written to a floppy or used to create a bootable CD-ROM.

Intended Audience

The audience for this book consists of two groups. The first group is practitioners who want to learn about the internals of Linux or Inferno, as well as reinforce their understanding of the basic principles. People in this group will likely come to the book with substantial exposure and experience in their respective OS. They will also likely have some familiarity with some of the concepts and techniques of OS operation. There is a possibility that such readers might not have been exposed to some of the data structures discussed in this book. Any book on data structures will provide the necessary background. Likewise, books on computer organization can provide good background in computer hardware. For this group, the chapters on general principles will help fill in any gaps in their knowledge, and the respective chapters on Linux and Inferno will provide introductions to the internals of those operating systems.

The second group this book serves is instructors and students of operating systems classes. Both introductory and advanced OS courses can make use of this book. Typical prerequisites for OS courses include a data structures course and a computer organization course. Some sections of this book assume that kind of background. Other sections are connected to programming languages, their compilers, and their run-time environments. Although courses on programming languages and compilers are helpful background, they are not necessary to study this material.

Using the Book

A book is intended to be read straight through, from start to finish, in most cases. This book can be used in that way. However, most instructors will not use it in that way. Rather, the most effective way to use this book in the classroom is to take selected material from each of the major parts. The full range of material provides each instructor with the flexibility to select the material that best fits the style of the course and personal preferences.

An instructor of an operating system course will choose those sections that support the course's general pedagogical approach. For example, one approach might focus on concepts and techniques in the abstract. Coverage of the difficulties associated with implementing those techniques on real hardware is traded off for theoretical depth. Another approach might trade off time used to present general principles for time used to illustrate the application of principles through a survey of a number of real operating systems. The final common approach to introductory classes combines a study of general principles with an in-depth examination of the implementation of an OS. In this last type of course, students are often required to make modifications to the operating system they study. Such in-depth experience with the internals of an existing OS is also often found in advanced operating systems classes.

Now, consider a set of recommended chapters and sections for each style of course. For general principles courses, the focus should be on a thorough coverage of Chapters 1, 5, 9, 13, 17, 21, and 22. Instructors might want to supplement this material with selected examples from Chapters 2, 6, 10, 14, and 18. Similarly, these chapters can be assigned as outside readings. One particular formulation of this type of course deserves special attention. The course designated CS220 in the Computing Curriculum 2001 specifies a number of specific topics that should be covered. The following sections provide good coverage of those recommendations: 1.1–1.4, 1.6, 5.1–5.8, 9.1–9.3, 9.5–9.6, 13.1–13.4, 13.7.1, 17.1–17.4, 17.7, 21.1–21.2, 21.4–21.5, 22.1, and 22.3–22.4. Of course, instructors are not restricted to covering only these sections. Material in other sections and chapters can be included to supplement the CC2001 recommendations.

If the principles-and-survey approach is used, Chapters 2, 6, 10, 14, and 18 should be in the classroom presentation. If the additional material places too great a demand on classroom time, instructors can be selective about which topics from the principles chapters to present. For example, the proof of the optimality of shortest job first and the formalism for name spaces can be safely omitted. Other examples of existing operating systems, not included here, make for good outside reading assignments.

It is also quite common to structure operating systems courses with a "hands-on" component. In this approach, students are generally expected to familiarize themselves with and make modifications to an existing OS. Frequently, the OS that is used is a relatively small one to make the expectations more manageable. This book also provides material that supports this style of course. The instructor can choose between two existing operating systems: Inferno and Linux. Instructors using Inferno should cover Chapters 3, 7, 11, 15, and 19 in addition to the general principles. Similarly for Linux, Chapters 4, 8, 12, 16, and 20 should be included. In covering the chapters that present code, it is important not to present too much code in the classroom. Experience has shown that it doesn't take long for all code to start looking the same, and the additional benefit dwindles. It is better to present a few smaller bits that illustrate particularly important points and have the students learn the rest with outside reading and assignments. The ability to read and understand real code is a valuable benefit of this type of organization.

The last curriculum example to consider is that of an advanced operating systems course. Considering the nature of advanced courses, several parts of the book could be used well. If the students have come from an introductory course that did not cover all the principles discussed here, then classroom time could be spent presenting them and digging more deeply into any of the principles. Similarly, the survey chapters are a good launching point for a more thorough examination of any one of them or for a more encompassing survey of real operating systems. Finally, for those students whose introductory course did not include experience with operating system internals, the detailed coverage of Inferno and Linux provides a starting place for such experience. These various curriculum designs are summarized in the following table:

Chapter	Principles Only	Principles and Examples	Principles and Inferno	Principles and Linux
1	$\surd$	$\surd$	$\surd$	$\surd$
2	•	$\surd$		
3			$\surd$	
4				$\surd$
5	$\surd$	$\surd$	$\surd$	$\surd$
6	•	$\surd$		
7			$\surd$	
8				$\surd$
9	$\surd$	$\surd$	$\surd$	$\surd$
10	•	$\surd$		
11			$\surd$	
12				$\surd$
13	$\surd$	$\surd$	$\surd$	$\surd$
14	•	$\surd$		
15			$\surd$	
16				$\surd$
17	$\surd$	$\surd$	$\surd$	$\surd$
18	•	$\surd$		
19			$\surd$	
20				$\surd$
21	$\surd$	$\surd$	$\surd$	$\surd$
22	$\surd$	$\surd$	$\surd$	$\surd$

$\surd$: The chapter is covered.
• : Selected topics from the chapter are covered.

Features of the Book

Each division of the book presents a topic from several perspectives: general principles, survey of applications, and detailed design and implementation of Inferno and Linux. The general-principles chapters include a number of key features. Several techniques are presented in the form of semiformal algorithms that are suitable for implementation. These algorithms are set apart typographically. In a number of cases, techniques are illustrated with detailed examples, also with distinct formatting. Finally, these chapters include a number of historical notes to help establish context. In those chapters that present detailed discussions of Inferno and Linux, I focus on relatively small parts of the kernel that illustrate the techniques and principles covered in the principles chapters. Each function I present is broken down in to small fragments, and I describe each of those in some detail. The result is a detailed study of some key elements of the respective kernels. These chapters also include exercises that ask the student to "get their hands dirty" making changes to Inferno and to Linux. In addition to these general features, here are some of the key topics discussed:

- *Chapter 1*: background, history, organization, bootstrapping, and system calls
- *Chapters 3, 4*: Inferno and Linux history, structure, initialization, and system calls
- *Chapter 5*: process representation, process scheduling, context switching, mutual exclusion, and deadlock
- *Chapters 7, 8*: process representation, creation, and scheduling in Inferno and Linux, including the new $O(1)$ scheduler in Linux
- *Chapter 9*: address translation techniques, variable-sized allocation techniques (including a comparative example), swapping, and paging
- *Chapter 11*: pool/block allocation and garbage collection in Inferno
- *Chapter 12*: zone/slab allocation, page tables, and page faults in Linux
- *Chapter 13*: overview of I/O hardware, techniques for controlling devices, and selected device management techniques
- *Chapter 15*: device driver structure, parallel port driver, keyboard driver, and IDE disk driver in Inferno
- *Chapter 16*: two-half interrupt handler, parallel port driver, and floppy disk driver in Linux
- *Chapter 17*: name spaces, storage management techniques, and journaled file systems
- *Chapter 19*: Inferno file server design, the Styx protocol, and the kfs file system
- *Chapter 20*: the Linux Virtual File System and the EXT3 file system
- *Chapter 21*: basic security techniques and threats, the Orange Book, encryption, and the Multics protection rings
- *Chapter 22*: resource sharing, synchronous operation, clusters, grids, distributed clocks, and election algorithms

Operating System Examples

The two operating systems discussed in detail in this book each provide their own advantages. Inferno is a relatively small operating system, making its details easier to grasp. Inferno is also somewhat unique in that it was designed to run not only as a conventional native operating system, but also as an application running on a host operating system. This hosted capability makes it significantly easier for students to install the OS on their own machines. It also simplifies the process of testing new versions as students debug their assignments. Linux, on the other hand, is a very familiar system—much more so than Inferno. Consequently, studying it provides the student with more directly applicable experience. It also provides examples of some of the more complex techniques not found in Inferno.

The version of Inferno used in this book is the release of May 10, 2007. The most recent distribution of Inferno can be found at the Vita Nuova Web site, *http://www.vitanuova.com*. Current development and recent revision history can be found on Google's code-hosting site at *http://code.google.com/p/inferno-os/*. I discuss the process of compiling and running Inferno in the appendices.

I use version 2.6.18 of the Linux kernel here. The primary site for both current and older versions of the Linux kernel is *http://www.kernel.org*. Two excellent sources of information on building a Linux kernel can be found in the "Linux Kernel HOWTO" by Brian Ward and the "Kernel Rebuild Guide" by Kwan Lowe.

Some of the source code for examples in Chapters 2, 6, 10, 14, and 18 is also available on the Web. The full source code for the CTSS operating system can be found at *http://www.piercefuller.com/library/ctss.html*. Some portions of the source code to Multics can be found at *http://www.multicians.org*. Old versions of UNIX, including the sixth edition, can be found at *http://www.tuhs.org*. The 4.3BSD version of UNIX is being maintained by the International Free Computing Task Force (IFCTF) as 4.3BSD-Quasijarus. The home for this project is *http://ifctfvax.harhan.org/Quasijarus* where these updates, as well as older versions, are all available. The primary resource for TinyOS is *http://www.tinyos.net*. Finally, the home for the Xen virtual machine monitor is *http://www.cl.cam.ac.uk/research/srg/netos/xen*.

Source Code Formatting

The source code fragments in this book are formatted using Knuth and Levy's CWEB system of structured documentation. Keywords and data types are typeset in boldface. Identifiers not in all caps are typeset in italic; identifiers in all caps are typeset in a monospace font. Several of the C language operators are multicharacter sequences that use characters from the ASCII character set. When presented here, some of these are replaced by common mathematical symbols, which in some cases express the meaning more directly. The correspondence between the C operators in ASCII and the symbols typeset in this text are summarized in the following table:

ASCII	Symbol
`->`	$\rightarrow$
`NULL`	Λ
`~`	$\sim$
`^`	$\oplus$
`==`	$\equiv$
`!=`	$\neq$
`>=`	$\geq$
`<=`	$\leq$
`!`	$\neg$
`&&`	$\land$
`||`	$\lor$
`>>`	$\gg$
`<<`	$\ll$

In addition to these conventions, `CWEB` formats octal and hexadecimal constants differently. An octal constant that would be expressed as `0123` in ASCII is typeset as $^{\circ}\mathit{123}$, and the hexadecimal constant `0x123` is typeset as $^{\#}$`123`.

It might seem strange that our printed representation doesn't appear in the same form as the compiler input that is typed in a text editor. Although today we don't see this sort of difference much outside of `WEB` and `CWEB`, it has a long tradition. A number of languages allowed variation in the characters used for operators and such because not all installations used the same character sets. Some environments had very limited character sets with some not even including lowercase letters. Others had extremely rich character sets, allowing programmers to directly enter mathematical symbols, such as those we use here for "not equal to," "less than or equal to," and "greater than or equal to." C itself defines trigraphs to allow for its use in environments that do not include all needed characters. Beyond that, it has been quite common for published code to have a different look to it, much as typeset text has a different look from the output of a typewriter. In that spirit, Algol 68 formalized the difference between the published form, called the strict language, and the compiler-input form, called the reference language. It is interesting to note that Algol was one of the languages that influenced the design of C, and the typesetting practices used with Algol influenced Knuth's development of `WEB`. The two lines of influence have converged in `CWEB`, whose printed representation I use here.

Supplementary Material

Supporting material for this book can be found on the Course Technology Web site at *http://www.course.com*. This same material is included on the instructor's CD, which can be obtained from a Course Technology sales representative. Copies of the relevant versions of Inferno and Linux are included with the supplementary material. The material also includes solutions to most of the exercises. Presentation material is included as well.

Acknowledgments

No project of this magnitude can be completed without help. There are numerous people whose support, encouragement, and assistance have been invaluable. I would first like to thank my wife Mary and my daughter Rachel. They have put up with innumerable nights of my frustration and of losing me to my office. But they never stopped believing in me or in this project. They picked up the slack around the house as I wrote. The interest and support of my other family and friends have also been a much welcome source of encouragement.

Next, I would like to thank my colleagues, both at FedEx and at the University of Memphis. No matter how hard an author tries to manage time (or at least no matter how hard *I* try to manage time), writing a book like this affects other responsibilities. I would particularly like to express my appreciation to Miley Ainsworth of FedEx Labs for his support of this project. I would also like to thank my colleague at FedEx, Tim Gregory, for his assistance in understanding TinyOS and for reviewing what I wrote about it.

The next group of people I'd like to thank is all of my operating systems students over the past few years. They have been my test subjects for this book. They have made

do with partial drafts, typos, and awkward grammar. Nevertheless, many of them have expressed support and encouragement for the project. I'd like to recognize a few students who provided helpful feedback. They include Bob Bradley, Jim Greer, Taliesin Penfound, and Debbie Travis.

I have also corresponded with several members of the operating systems community who provided valuable assistance and feedback. They are Charles Forsyth, of Vita Nuova Holdings Ltd.; Tom Van Vleck, former member of the Multics development team and maintainer of the multicians.org Web site; Stephen Hoffman, of the HP OpenVMS group; and Digby Tarvin, of the University of New South Wales. I would also like to thank Jim Aman at Saint Xavier University, Charles Anderson at Western Oregon University, Bob Bradley at the University of Tennessee at Martin, Thomas Liu at New Jersey City University, Chung-Horng Lung at Carleton University, Jon Weissman at the University of Minnesota, and Dongyan Xu at Purdue University. Their careful review of the early drafts led to substantial improvements in this material.

Finally, I want to express my appreciation to everyone at Course Technology. There's no way to list everyone who has contributed to this project. However, a few names must be recognized. First, I'd like to thank Mary Franz, who first saw the potential for this book and helped me through the early stages of the process. There are three people who have been there working with me all the way through the project. I am supremely grateful to Alyssa Pratt, Jim Markham, and Matt Hutchinson for everything they've done to make this book possible. Everyone's input and involvement have made this a better book. Any remaining flaws are entirely my own. I can't thank everyone enough for all the support in making this a reality.

Brian L. Stuart