Principles of Computer System Design

An Introduction

Part II
Chapters 7–11

Jerome H. Saltzer

M. Frans Kaashoek

Massachusetts Institute of Technology

Version 5.0

Suggestions, Comments, Corrections, and Requests to waive license restrictions: Please send correspondence by electronic mail to:

Saltzer@mit.edu

and

kaashoek@mit.edu

Contents

PART I [In Printed Textbook]

Part II [On-Line]

List of Sidebars

PART I [In Printed Textbook]

CHAPTER 1 Systems

CHAPTER 2 Elements of Computer System Organization

CHAPTER 3 The Design of Naming Schemes

CHAPTER 4 Enforcing Modularity with Clients and Services

CHAPTER 5 Enforcing Modularity with Virtualization

CHAPTER 6 Performance

Part II [On-Line]

CHAPTER 7 The Network as a System and as a System Component

CHAPTER 8 Fault Tolerance: Reliable Systems from Unreliable Components

CHAPTER 9 Atomicity: All-or-Nothing and Before-or-After

CHAPTER 10 Consistency

CHAPTER 11 Information Security

Preface to Part II

This textbook, *Principles of Computer System Design: An Introduction,* is an introduction to the principles and abstractions used in the design of computer systems. It is an outgrowth of notes written by the authors for the M.I.T. Electrical Engineering and Computer Science course 6.033, Computer System Engineering, over a period of 40-plus years.

The book is published in two parts:

- Part I, containing chapters 1-6 and supporting materials for those chapters, is a traditional printed textbook published by Morgan Kaufman, an imprint of Elsevier. (ISBN: 978–012374957–4)
- Part II, consisting of Chapters 7–11 and supporting materials for those chapters, is made available on-line by M.I.T. OpenCourseWare and the authors as an open educational resource.

Availability of the two parts and various supporting materials is described in the section with that title below.

Part II of the textbook continues a main theme of Part I—enforcing modularity—by introducing still stronger forms of modularity. Part I introduces methods that help prevent accidental errors in one module from propagating to another. Part II introduces stronger forms of modularity that can help protect against component and system failures and against malicious attacks. Part II explores communication networks, constructing reliable systems from unreliable components, creating all-or-nothing and before-or-after transactions, and implementing security. In doing so, Part II also continues a second main theme of Part I by introducing several additional design principles related to stronger forms of modularity.

A detailed description of the contents of the chapters of Part II can be found in Part I, in the section "About Part II" on page 369. Part II also includes a table of contents for both Parts I and II, copies of the Suggested Additional Readings and Glossary, Problem Sets for both Parts I and II, and a comprehensive Index of Concepts with page numbers for both Parts I and II in a single alphabetic list.

Availability

The authors and MIT OpenCourseWare provide, free of charge, on-line versions of Chapters 7 through 11, the problem sets, the glossary, and a comprehensive index. Those materials can be found at

`http://ocw.mit.edu/Saltzer-Kaashoek`

in the form of a series of PDF files (requires Adobe Reader), one per chapter or major supporting section, as well as a single PDF file containing the entire set.

The publisher of the printed book also maintains a set of on-line resources at

`www.ElsevierDirect.com/9780123749574`

Click on the link "Companion Materials", where you will find Part II of the book as well as other resources, including figures from the text in several formats. Additional materials for instructors (registration required) can be found by clicking the "Manual" link.

There are two additional sources of supporting material related to the teaching of course 6.033 Computer Systems Engineering, at M.I.T. The first source is an OpenCourseWare site containing materials from the teaching of the class in 2005: a class description; lecture, reading, and assignment schedule; board layouts; and many lecture videos. These materials are at

`http://ocw.mit.edu/6-033`

The second source is a Web site for the current 6.033 class. This site contains the currernt lecture schedule which includes assignments, lecturer notes, and slides. There is also a thirteen-year archive of class assignments, design projects, and quizzes. These materials are all at

`http://mit.edu/6.033`

(Some copyrighted or privacy-sensitive materials on that Web site are restricted to current MIT students.)

Acknowledgments

This textbook began as a set of notes for the advanced undergraduate course Engineering of Computer Systems (6.033, originally 6.233), offered by the Department of Electrical Engineering and Computer Science of the Massachusetts Institute of Technology starting in 1968. The text has benefited from some four decades of comments and suggestions by many faculty members, visitors, recitation instructors, teaching assistants, and students. Over 5,000 students have used (and suffered through) draft versions, and observations of their learning experiences (as well as frequent confusion caused by the text) have informed the writing. We are grateful for those many contributions. In addition, certain aspects deserve specific acknowledgment.

1. Naming (Section 2.2 and Chapter 3)

The concept and organization of the materials on naming grew out of extensive discussions with Michael D. Schroeder. The naming model (and part of our development) follows closely the one developed by D. Austin Henderson in his Ph.D. thesis. Stephen A. Ward suggested some useful generalizations of the naming model, and Roger Needham suggested several concepts in response to an earlier version of this material. That earlier version, including in-depth examples of the naming model applied to addressing architectures and file systems, and an historical bibliography, was published as Chapter 3 in Rudolf Bayer et al., editors, *Operating Systems: An Advanced Course*, *Lecture Notes in Computer Science 60*, pages 99–208. Springer-Verlag, 1978, reprinted 1984. Additional ideas have been contributed by many others, including Ion Stoica, Karen Sollins, Daniel Jackson, Butler Lampson, David Karger, and Hari Balakrishnan.

2. Enforced Modularity and Virtualization (Chapters 4 and 5)

Chapter *4* was heavily influenced by lectures on the same topic by David L. Tennenhouse. Both chapters have been improved by substantial feedback from Hari Balakrishnan, Russ Cox, Michael Ernst, Eddie Kohler, Chris Laas, Barbara H. Liskov, Nancy Lynch, Samuel Madden, Robert T. Morris, Max Poletto, Martin Rinard, Susan Ruff, Gerald Jay Sussman, Julie Sussman, and Michael Walfish.

3. Networks (Chapter 7[on-line])

Conversations with David D. Clark and David L. Tennenhouse were instrumental in laying out the organization of this chapter, and lectures by Clark were the basis for part of the presentation. Robert H. Halstead Jr. wrote an early draft set of notes about networking, and some of his ideas have also been borrowed. Hari Balakrishnan provided many suggestions and corrections and helped sort out muddled explanations, and Julie Sussman and Susan Ruff pointed out many opportunities to improve the presentation. The material on congestion control was developed with the help of extensive discussions with Hari Balakrishnan and Robert T. Morris, and is based in part on ideas from Raj Jain.

4. Fault Tolerance (Chapter 8[on-line])

Most of the concepts and examples in this chapter were originally articulated by Claude Shannon, Edward F. Moore, David Huffman, Edward J. McCluskey, Butler W. Lampson, Daniel P. Siewiorek, and Jim N. Gray.

5. Transactions and Consistency (Chapters 9[on-line] and 10[on-line])

The material of the transactions and consistency chapters has been developed over the course of four decades with aid and ideas from many sources. The concept of version histories is due to Jack Dennis, and the particular form of all-or-nothing and before-or-after atomicity with version histories developed here is due to David P. Reed. Jim N. Gray not only came up with many of the ideas described in these two chapters, he also provided extensive comments. (That doesn't imply endorsement—he disagreed strongly about the importance of some of the ideas!) Other helpful comments and suggestions were made by Hari Balakrishnan, Andrew Herbert, Butler W. Lampson, Barbara H. Liskov, Samuel R. Madden, Larry Rudolph, Gerald Jay Sussman, and Julie Sussman.

6. Computer Security (Chapter 11[on-line])

Sections *11.1* and *11.6* draw heavily from the paper "The Protection of Information in Computer Systems" by Jerome H. Saltzer and Michael D. Schroeder, *Proceedings of the IEEE 63,* 9 (September, 1975), pages 1278–1308. Ronald Rivest, David Mazières, and Robert T. Morris made significant contributions to material presented throughout the chapter. Brad Chen, Michael Ernst, Kevin Fu, Charles Leiserson, Susan Ruff, and Seth Teller made numerous suggestions for improving the text.

7. Suggested Outside Readings

Ideas for suggested readings have come from many sources. Particular thanks must go to Michael D. Schroeder, who uncovered several of the classic systems papers in places outside computer science where nobody else would have thought to look, Edward D. Lazowska, who provided an extensive reading list used at the University of Washington, and Butler W. Lampson, who provided a thoughtful review of the list.

8. The Exercises and Problem Sets

The exercises at the end of each chapter and the problem sets at the end of the book have been collected, suggested, tried, debugged, and revised by many different faculty members, instructors, teaching assistants, and undergraduate students over a period of 40 years in the process of constructing quizzes and examinations while teaching the material of the text.

Certain of the longer exercises and most of the problem sets, which are based on lead-in stories and include several related questions, represent a substantial effort by a single individual. For those problem sets not developed by one of the authors, a credit line appears in a footnote on the first page of the problem set.

Following each problem or problem set is an identifier of the form "*1978–3–14*". This identifier reports the year, examination number, and problem number of the examination in which some version of that problem first appeared.

Jerome H. Saltzer
M. Frans Kaashoek
2009

Computer System Design Principles

Throughout the text, the description of a design principle presents its name in a **boldfaced** display, and each place that the principle is used highlights it in <u>*underlined italics*</u>.

Design principles applicable to many areas of computer systems

- **Adopt sweeping simplifications**
  So you can see what you are doing.
- **Avoid excessive generality**
  If it is good for everything, it is good for nothing.
- **Avoid rarely used components**
  Deterioration and corruption accumulate unnoticed—until the next use.
- **Be explicit**
  Get all of the assumptions out on the table.
- **Decouple modules with indirection**
  Indirection supports replaceability.
- **Design for iteration**
  You won't get it right the first time, so make it easy to change.
- **End-to-end argument**
  The application knows best.
- **Escalating complexity principle**
  Adding a feature increases complexity out of proportion.
- **Incommensurate scaling rule**
  Changing a parameter by a factor of ten requires a new design.
- **Keep digging principle**
  Complex systems fail for complex reasons.
- **Law of diminishing returns**
  The more one improves some measure of goodness, the more effort the next improvement will require.
- **Open design principle**
  Let anyone comment on the design; you need all the help you can get.
- **Principle of least astonishment**
  People are part of the system. Choose interfaces that match the user’s experience,

expectations, and mental models.

- **Robustness principle**
  Be tolerant of inputs, strict on outputs.
- **Safety margin principle**
  Keep track of the distance to the edge of the cliff or you may fall over the edge.
- **Unyielding foundations rule**
  It is easier to change a module than to change the modularity.

Design principles applicable to specific areas of computer systems

- *Atomicity:* **Golden rule of atomicity**
  Never modify the only copy!
- *Coordination:* **One-writer principle**
  If each variable has only one writer, coordination is simpler.
- *Durability:* **The durability mantra**
  Multiple copies, widely separated and independently administered.
- *Security:* **Minimize secrets**
  Because they probably won't remain secret for long.
- *Security:* **Complete mediation**
  Check every operation for authenticity, integrity, and authorization.
- *Security:* **Fail-safe defaults**
  Most users won't change them, so set defaults to do something safe.
- *Security:* **Least privilege principle**
  Don't store lunch in the safe with the jewels.
- *Security:* **Economy of mechanism**
  The less there is, the more likely you will get it right.
- *Security:* **Minimize common mechanism**
  Shared mechanisms provide unwanted communication paths.

Design Hints (useful but not as compelling as design principles)

- **Exploit brute force**
- **Instead of reducing latency, hide it**
- **Optimize for the common case**
- **Separate mechanism from policy**