Topics in the Engineering of Computer Systems

Chapter Nine: Suggestions for Further Reading

Jerome H. Saltzer

April, 1996

Table of Contents

Introduction

The hardware technology that underlies computer systems has improved so rapidly and continuously for more than four decades that the ground rules for system design are constantly subject to change. It takes many years for knowledge and experience to be compiled, digested, and presented in the form of a book, so books about computer systems often seem dated or obsolete by the time they appear in print. Even though some underlying principles are unchanging, the rapid obsolescence of details acts to discourage prospective book authors, and as a result some important ideas never do get well-documented in books. For this reason, an essential part of the study of computer systems is found in current—and sometimes older—technical papers, professional journal articles, research reports, and occasional, unpublished memoranda that circulate among active workers in the field.

Despite that caveat, there are a few books, relatively recent additions to the literature in computer systems, that are worth having on the shelf. Up until the mid-1980's the books that existed were for the most part commissioned by textbook publishers to fill a market and they tended to emphasize the mechanical aspects of systems rather than insight into their design. Starting around 1985, however, several very good books started to appear, when a number of professional system designers became inspired to capture their insights. The appearance of these books also suggests that the concepts involved in computer system design are finally beginning to stabilize a bit. (Or it may just be that computer system technology is beginning to shorten the latencies involved in book publishing.)

The heart of the computer systems literature is found in published papers. Two of the best sources are Association for Computing Machinery (ACM) publications: the journal *ACM Transactions on Computer Systems (TOCS)* and the bi-annual series of conference proceedings, the *ACM Symposium on Operating Systems Principles (SOSP).* The best papers of each SOSP are published in a following issue of TOCS, and the rest—in recent years all—of the papers of each symposium appear in a special edition of *Operating Systems Review*, an ACM special interest group quarterly that publishes a fifth issue in symposium years. One should not, however, assume that these are the only sources—worthwhile papers about computer systems appear in a wide variety of other journals.

On the following pages will be found a set of suggestions for further reading about computer systems, both papers and books. These readings have been selected to emphasize the best available thinking, best illustrations of problems, and most interesting case studies of computer systems. The readings have been reviewed for obsolescence, but it is often the case that a good idea is still best described by a paper from some time ago, where the idea was developed in a context that no longer seems very interesting. Sometimes that early context is much simpler than today's systems, thus making it easier to see how the idea works. Often, an early author was the first on the scene, so it was necessary to describe things more completely than do modern authors who usually assume significant familiarity with the surroundings. Thus the older readings included here provide a very useful complement to current works.

By its nature, the study of the engineering of computer systems overlaps with other areas of computer science, particularly computer architecture, programming languages, data bases, information retrieval, and data communications. Each of those areas has an extensive literature of its own, and it is often not obvious where to draw the boundary lines. As a general rule, this reading list tries to provide only first-level guidance on where to get started in those related areas.

One thing the reader must watch for is that the terminology of the computer systems field is not agreed upon, so the literature is often confusing even to the professional. In addition, the quality level of the literature is quite variable, ranging from the literate through the readable to the barely comprehensible. Although the

selections here try to avoid the completely incomprehensible, the reader must still be prepared for some papers, however important in their content, that do not explain their subject as well as they could.

In the material that follows, each citation is accompanied by a comment suggesting why that paper is worth reading—its importance, interest, and relation to other readings. When a single paper serves more than one area of interest, cross-references appear rather than multiple citations.

1. The systems bookshelf

As mentioned above, a few wonderful and several really good books about computer systems have recently begun to appear. Here are the must-have items for the reference shelf of the computer systems designer. In addition to these books, the later groupings of readings by topic include other books, generally of narrower interest.

1.1 Wonderful books.

1.1.1 David A. Patterson and John L. Hennessy. *Computer Organization and Design: The hardware/software interface.* Morgan Kaufman, 1994. ISBN: 1-55860-281-X. 648 + various pages. The title page gives the authors' names in the opposite order.

1.1.2 David A. Patterson and John L. Hennessy. *Computer Architecture: A Quantitative Approach.* Morgan Kaufman, Second edition, 1996. ISBN: 1-58860-329-8. 760 + various pages. The cover gives the authors' names in the opposite order.

This pair of books together provide a spectacular tour-de-force that explores much of the design space of current computer architecture. The first book provides the fundamentals, the second provides depth. One of the best features is that each area includes a discussion of misguided ideas and their pitfalls. Even though the subject matter gets very sophisticated, the books are always very readable. The books are opinionated (with a strong bias toward RISC architecture), but nevertheless this is a definitive work on computer organization from the system perspective.

1.1.3 Daniel P. Siewiorek, C. Gordon Bell, and Allen Newell, editors. *Computer Structures: Principles and Examples.* McGraw-Hill, 1982. ISBN: 0-07-057302-6. 926 pages.

An important standard reference book, reprinting original papers that describe practically every machine architecture of interest as of 1982. One pines for an update that preserves all that is here, but that adds the design ideas of another decade or so. If you look for this book in a library, beware of the similarly titled *Computer Structures: Readings and Examples,* also by Bell and Newell, but eleven years older.

1.1.4 Raj Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991. ISBN 0-471-50336-3. 685 pages.

Much work on performance analysis of computer systems originates in academic settings and focuses on analysis that is mathematically tractable rather than on measurements that matter. This book is at the other end of the spectrum. It is written by someone with extensive industrial experience but an academic flair for explaining things. If you have a real performance analysis problem, it will tell you how to tackle it, how to avoid measuring the wrong thing, and how to step by other pitfalls.

1.1.5 Jim [N.] Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques.* Morgan Kaufmann, San Mateo, California, 1993 (Look for the low-bulk paper edition, which became available with the third printing in 1994). ISBN: 1-55860-190-2. 1070 pages.

All aspects of fault-tolerance, atomicity, coordination, recovery, rollback, logs, locks, transactions, and engineering trade-offs for performance are pulled together in this comprehensive book. This is the definitive work on transactions. Although not intended for beginners, the quality of its explanations is very high, making this complex material surprisingly accessible. Excellent glossary of terms. The historical notes are good as far as they go, but are somewhat database-centric and should not be taken as the final word.

1.1.6 Bruce Schneier. *Applied Cryptography.* John Wiley and Sons, second edition, 1996. ISBN: 0-471-11709-9 (paperback). 758 pages.

Everything you might need to know to put cryptography to work in practical computing situations, including a well-balanced perspective on what works and what doesn't. Saves the need to read and sort through the thousand or so technical papers on the subject. Protocols, techniques, algorithms, real-world considerations, and source code can all be found here. In addition to being competent, it is also entertainingly written and very articulate.

1.1.7 Frederick P. Brooks, Jr. *The Mythical Man-Month.* Addison-Wesley, 20th Anniversary edition, 1995 (First edition 1974, reprinted with corrections, 1982). ISBN: 0-201-83595-9 (paperback). 322 pages.

Well-written, full of insight. This reading is by far the most significant one on the subject of controlling system development. This is where you learn why adding more staff to a project that is behind schedule will delay it further. Although a few of the chapters are now a bit dated, much of the material here is timeless. Trouble in system development is also timeless, as evidenced by continual reports of failures of large system projects. Most successful system designers have a copy of this book on their bookshelf, and some claim to reread it at least once a year. Most of the 1995 edition is identical to the 1974 edition; the newer edition adds Brooks' *No Silver Bullets* paper (which is well worth reading), and some summarizing chapters.

1.1.8 Alan F. Westin. *Privacy and Freedom*. Atheneum Press, 1967. 487 pages.

If you have any interest in privacy, this book is the place to start. It is the comprehensive treatment, by a constitutional lawyer, of what privacy is, why it matters, and its position in the U.S. legal framework.

1.2 Really good books.

1.2.1 Henry Petroski. *Design Paradigms: Case Histories of Error and Judgment in Engineering.* Cambridge University Press, 1994. ISBN: 0-521-46108-1 (hardcover), 0-521-46649-0 (paperback). 209 pages.

This remarkable book explores what in the mind-set of the designers (in the examples, civil engineers,) allowed them to make what in retrospect were massive design errors. The failures analyzed range from the transportation of columns in Rome through the 1982 collapse of the walkway in the Kansas City Hyatt Regency hotel, with a number of famous bridge collapses in between. Petroski analyzes particularly well how a failure of a scaled-up design often reveals that the original design worked correctly, but for a different reason than originally thought. There is no mention of computer systems in this book, but it contains many lessons for computer system designers.

1.2.2 Andrew S. Tanenbaum. *Modern Operating Systems.* Prentice-Hall, 1992. ISBN 0-13-588187-0 (hardcover); 0-13-595752-4 (paperback). 728 pages.

A thorough tutorial introduction to the world of operating systems. Tends to emphasize the mechanics; insight into why things are designed the way they are is there but in many cases requires teasing out. But for a starting

point, it is filled with street knowledge that is needed to get into the rest of the literature. Includes useful case studies of MS-DOS, Unix, and Mach, as well as the author's own system, Amoeba.

1.2.3 Andrew S. Tanenbaum. *Computer Networks.* Prentice-Hall, third edition, 1996. ISBN: 0-13-349945-6. 813 pages.

A thorough tutorial introduction to the world of networks. Like the same author's book on operating systems, this one also tends to emphasize the mechanics. But again it is a storehouse of up-to-date street knowledge, this time about computer communications, that is needed to get into (or perhaps avoid the need to consult) the rest of the literature. The book includes a selective and thoughtfully annotated bibliography on computer networks.

1.2.4 Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik. *Quantitative System Performance: Computer System Analysis using Queuing network models*. Prentice-Hall, 1984. ISBN: 0-13-746975-6. 417 pages.

This is one of the books on performance analysis that originates in an academic setting, but it is better than most, because it emphasizes operational analysis rather than stochastic analysis. Queuing network models are less important inside computer systems these days than they were at the time the book was being written, but they can still be very useful in analyzing networks constructed of lots of interacting servers.

1.2.5 Radia Perlman. *Interconnections: bridges and routers*. Addison-Wesley, 1992. ISBN: 0-201-56332-0. 389 pages.

Everything you could possibly want to know about how the network layer actually works. The style is engagingly informal, but the content is absolutely first-class, and every possible variation is explored

1.2.6 John E. McNamara. *Technical Aspects of Data Communications.* Digital Press, Bedford, Massachusetts, third edition, 1988. ISBN: 1-55558-007-6. 383 pages.

A handbook filled with useful facts about commonly-encountered data communication hardware and facilities: how asynchronous lines and modems work, the specifications of RS-232, RS-422, X.20, X.21, CRC details, channel conditioning, etc. Replaces a whole file cabinet full of hard-to-find facts.

1.2.7 William Stallings. *Local and Metropolitan Networks.* Macmillan, fourth edition, 1993. ISBN: 0-02-415465-2. 550 pages.

A very broad and fast-moving treatment that provides a good overview of local network technology. Updated frequently, this is the fourth edition in 9 years. In addition to the expected coverage of the usual local area network technologies, there are chapters on circuit-switched branch exchanges, on higher speed technologies, and on internetwork interconnection. Contains an up-to-date 15-page long bibliography, and each chapter ends with specific recommendations for further reading.

1.2.8 Simson Garfinkel. *PGP: Pretty Good Privacy.* O'Reilly & Associates, Sebastopol, California, 1995. ISBN: 1-56592-098-8 (paperback). 393 pages.

Nominally a user's guide to the PGP encryption package developed by Phil Zimmermann, this book starts out with six very readable overview chapters on the subject of encryption, its history, and the political and licensing environment that surrounds encryption systems. Even the later chapters, which give details on how to use PGP, are filled with interesting tidbits and advice that is actually applicable to all use of encryption.

1.2.9 David Burnham. *The rise of the computer state*. Random House, 1982. ISBN: 0-394-51437-8 (hardcover); 0-394-72375-9 (paperback). 273 pages.

Written by an experienced reporter who became interested in the implications of computer systems and networks on privacy, surveillance, and the balance of power between citizens and the government. Filled with illustrative examples and anecdotes. Despite its age, sufficiently forward-looking that it is still worth reading.

1.3. Other books deserving space on the systems bookshelf

Some books, while not quite qualifying for the "very good" category are still good enough that many computer systems professionals insist on having them on their bookshelves. In many cases the only objection to the book is that its appeal is narrow or that the book is centered in another area.

1.3.1. Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to algorithms.* MIT Press, 1990. 1028 pages. ISBN: 0-262-03141-8, 0-07-013143-0 (McGraw-Hill).

In a reading list on theory this book would almost certainly be in one of the higher categories. It appears in this list as an auxiliary item because occasionally a system designer needs an algorithm. This book is the place to find that algorithm together with the analysis necessary to decide whether or not it is appropriate for the application.

1.3.2. Philip A. Bernstein, Vassos Hadzilacos, and Nathan Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, 1987. ISBN 0-201-10715-5. 370 pages.

An aggressively theoretical and rigorous treatment of transactions, concurrency, recovery, atomicity, and replication. Anyone interested in contributing to the theory of these topics must start with this book, but system implementers may find it to be somewhat inaccessible.

1.3.3. Douglas K. Smith and Robert C. Alexander. *Fumbling the Future.* William Morrow and Company, 1988. ISBN 0-688-06959-2. 274 pages.

The history of the computing is littered with companies that attempted to add general-purpose computer systems to an existing business—Ford, Philco, Zenith, RCA, General Electric, Honeywell, A. T. & T., and Xerox are examples. None have succeeded, perhaps because when the going gets tough the option of walking away from this business is too attractive. This book documents how Xerox managed to snatch defeat from the jaws of victory by inventing the personal computer, then abandoning it.

1.3.4 Sape J. Mullender, editor. *Distributed Systems.* Addison-Wesley, second edition, 1993, reprinted with minor additions, 1995. ISBN: 0-201-62427-3. 601 pages.

This book is a collection of essays on various aspects of distributed system design, by many different authors. The materials are relatively new, and in some cases could use some more polishing. The result is a bit variable but the accumulation of ideas provides a useful resource pending the arrival of the definitive book on distributed systems.

1.3.5 Dimitri P. Bertsekas and Robert G. Gallager. *Data Networks.* Prentice-Hall, second edition, 1992. ISBN 0-13-200916-1. 556 pages.

An intense textbook on the theory that underlies packet-switched data communication networks. Not for casual browsing or for those easily intimidated by mathematics, but an excellent reference source for analysis.

1.3.6 Daniel P. Siewiorek and Robert S. Swarz. *Reliable Computer Systems: Design and Evaluation.* Digital Press, Bedford, Massachusetts, Second edition, 1992. ISBN 1-55558-075-0.

Probably the best comprehensive treatment of reliability that is available, with well-explained theory and reprints of several case studies from recent literature. Its only defect is a slight "academic" bias in that little judgement is expressed on alternative methods, and some examples are without warning of systems that were never really deployed. The first, 1982, edition, with the title *The Theory and Practice of Reliable System Design*, contains an almost completely different (but much older) set of case studies.

1.3.7 Samuel J. Leffler, Marshall Kirk McKusick, Michael J. Karels, and John S. Quarterman. *The Design and Implementation of the 4.3BSD Unix Operating System* Addison-Wesley, 1989. ISBN 0-201-06196-1; 0-201-54629-9.

This book provides a complete picture of the design and implementation of the Berkeley version of Unix. Well-written and full of detail.

1.3.8 Simson Garfinkel and Gene [Eugene H.] Spafford. *Practical Unix Security*. O'Reilly & Associates, Sebastopol, California, 1991 (Reprinted with corrections, 1991 and 1992). ISBN 0-937175-72-2 (paperback).

A really comprehensive guide to how to run a Unix system with some confidence that it is relatively safe against casual intruders. In addition to providing practical information for a system manager, it incidentally gives the reader quite a bit of insight into the style of thinking and design needed to provide security.

1.3.9 Dorothy E. R. Denning. *Cryptography and Data Security*. Addison-Wesley, 1982 (reprinted with corrections, 1983). ISBN: 0-201-10150-5. 400 pages.

Too-quick reading of the title might lead one to think that this book is only about cryptography, but it is actually two books in one—the "data security" part of the title refers to an excellent presentation of the state of the art (in 1982) in verification, kernel design, data base inference, and security policy systemization. Since not much has happened in those areas since then, this book remains a valuable addition to a system designer's library. The section on cryptography is also well-written, but if one is going to look at only one book on that topic, Schneier, reading 1.1.6 (see page 9-5), is the book of choice. The main flaw in this book is that it explains techniques that are proven by years of experience alongside other techniques that have never been tried in the field; there is little critical judgement as to which ideas are valuable, which are only promising, and which may prove to be crackpot.

1.3.10 Katie Hafner and John Markoff. Cyberpunk: outlaws and hackers on the computer frontier. Simon & Schuster (Touchstone), 1991. ISBN 0-671-68322-5 (hardcover), 0-671-77879-X (paperback). 368 pages.

A very readable yet thorough account of the scene surrounding three hackers: Kevin Mitnick, Hans Hubner (Pengo,) and Robert Tappan Morris. An example of a view from the media, but an unusually well-informed view.

1.3.11 Deborah G. Johnson and Helen Nissenbaum. *Computers, Ethics & Social Values*. Prentice-Hall, 1995. ISBN: 0-13-103110-4 (paperback). 714 pages.

A computer system designer is likely to consider reading a treatise on ethics to be a terribly boring way to spend the afternoon, and some of the papers in this extensive collection do match that stereotype. However, among the many scenarios, case studies, and other reprints in this volume are quite a number of interesting and thoughtful papers about the human consequences of computer system design. This collection is a good place to acquire the basic readings concerning privacy, risks, computer abuse, and software ownership as well as professional ethics in computer system design.

2 Generalities about systems

2.1 Ways of thinking about systems

There are quite a few books, especially by European academicians, that try to generalize the study of systems. They tend to be so abstract that it is hard to see how they apply to anything, so none of them are listed here. Instead, here are five old but surprisingly relevant papers that illustrate ways to think about systems. The areas touched are allometry, aerodynamics, hierarchy, ecology, and economics. These are followed by a series of more recent papers that simply illustrate the wide range of objectives for computer-based systems.

2.1.1 J[ohn] B[urdon] S[anderson] Haldane (1892-1964). On being the right size. In *Possible Worlds and Other Essays*, pages 20-28. Harper and Brothers Publishers, 1928. Also published by Chatto & Windus, London, 1927, and recently reprinted in John Maynard Smith, editor, *On Being the Right Size and Other Essays,* Oxford University Press, 1985. ISBN: 0-19-286045-3 (paperback) pages 1-8.

This is the classic paper that explains why a mouse the size of an elephant would collapse if it tried to stand up. Provides lessons on how to think about incommensurate scaling in all kinds of systems.

2.1.2 Alexander Graham Bell (1847-1922). The tetrahedral principle in kite structure. *National Geographic Magazine 14*, 6 (June, 1903) pages 219-251.

Another classic paper that demonstrates that arguments based on scale can be quite subtle. This paper—written at a time when physicists were still debating the theoretical possibility of building airplanes—describes the obvious scale argument against heavier-than-air craft and then demonstrates that one can increase scale of an airfoil in different ways and that the obvious scale argument does not apply to all those ways.

2.1.3 Herbert A. Simon (1916-). The architecture of complexity. *Proceedings of the American Philosophical Society 106*, 6 (December, 1962) pages 467-482. Republished as Chapter 4, pages 84-118, of *The Sciences of the Artificial,* M. I. T. Press, Cambridge, Massachusetts, 1969. ISBN: 0-262-191051-6 (hardcover) 0-262-69023-3 (paperback).

A tour-de-force of how hierarchy can be used as an organizing tool in dealing with complex systems. The examples are breathtaking in their range and scope—from watch-making and biology through political empires. The style of thinking shown in this paper suggests that it is not surprising that Simon later received a Nobel price in Economics.

2.1.4 LaMont C[ook] Cole (1916-1978). Man’s effect on nature. *The Explorer: Bulletin of the Cleveland Museum of Natural History 11,* 3 (Fall, 1969) pages 10-16.

This brief article looks at the Earth as a ecological system in which the effects of man lead both to surprises and to propagation of effects. It describes a classic example of propagation of effects: attempts to eliminate malaria in North Borneo led to an increase in the plague and roofs caving in.

2.1.5 Garrett [James] Hardin (1915-). The tragedy of the commons. *Science 162,* 3859 (December 13, 1968) pages 1243-1248.

A paper that explores a property of certain economic situations in which Adam Smith's “invisible hand” works against everyone's interest. Interesting for insight into how to predict things about otherwise hard-to-model systems.

2.2 Examples of objectives of systems

2.2.1 Alfred Spector and David [K.] Gifford. The space shuttle primary computer system. *Communications of the ACM 27,* 9 (September, 1984) pages 873-900.

The ground-control system for manned space missions. Emphasis is on continued operation, no matter what, once a mission starts.

2.2.2 David [K.] Gifford and Alfred Spector. The TWA reservation system. *Communications of the ACM 27,* 7 (July, 1984) pages 649-665.

A very big airline reservation and flight operations system. Airline reservations are among the most demanding of data base applications, and flight operations resembles military command and control in its objectives.

2.2.3 David [K.] Gifford and Alfred Spector. The Cirrus banking network. *Communications of the ACM 28,* 8 (August, 1985) pages 797-807.

Another paper in the series of question-and-answer case studies by Gifford and Spector, this time about the architecture of one of the networks that allow you to withdraw money from an ATM in Peoria even though your bank is in Boston.

2.2.4 Dennis M. Ritchie and Ken [L.] Thompson. The Unix time-sharing system. *Bell System Technical Journal 57,* 6, part 2 (1978) pages 1905-1930.

This paper describes an operating system with very low-key, but carefully-chosen and hard-to-discover objectives. Unix provides a hierarchical catalog structure, and succeeds in keeping naming completely distinct from file management. An earlier version of this paper appeared in the *Communications of the ACM 17*, 7 (July, 1974) pages 365-375, after being presented at the *Fourth ACM Symposium on Operating Systems Principles*. Unix evolved rapidly between 1973 and 1978, so the *BSTJ* version, though harder to find, contains significant additions, both in insight and in technical content.

2.2.5 T. D. Sterling and E. Laudon. Humanizing Information Systems. *Datamation 22*, 12 (December, 1976) pages 53-59.***

An old but reliable discussion of aspects of information systems that are dehumanizing and principles that can be used to avoid the problem.

3 Memory in Computer Systems

3.1 Memory Allocation

> In the area of memory allocation are two very old but still quite pertinent papers. Most recent work in allocation is on garbage collection for object-oriented languages. The garbage collection papers actually blend in to the topic of multilevel memory management. That area has a very extensive literature of its own, of which only a few key items are cited here.

3.1.1 Kenneth C. Knowlton. A fast storage allocator. *Communications of the ACM 8,* 10 (October, 1965) pages 623-624.

Original, and still best, description of a by-now widely-used technique for memory allocation that combines contiguous and block allocation in what is known as the “buddy system”.

3.1.2 Douglas T. Ross. The AED free storage package. *Communications of the ACM 10,* 8 (August, 1967) pages 481-492.

If you can sit still for the somewhat odd terminology of this paper, you will find that it contains seminal thinking on how to do storage allocation. Written from the perspective of an implementer of one of the first programming languages to make a serious attempt at handling storage allocation for its users.

3.1.3. Henry G. Baker, Jr. List processing in real time on a serial computer. *Communications of the ACM 21,* 4 (April, 1978) pages 280-294.

One of the most sophisticated storage allocation schemes to be documented: a copying garbage collector that avoids stopping the application for uncontrollably long periods. The approach used to model the algorithm and its load so as to predict maximum housekeeping time is quite interesting.

3.1.4 Henry Leiberman and Carl Hewitt. A real-time garbage collector based on the lifetimes of objects. *Communications of the ACM 26,* 6 (June, 1983) pages 419-429.

A refinement of Baker's garbage collector taking advantage of the observation that objects can be dynamically classified according to their probable lifetimes, and that longer-lived objects can be garbage-collected less often.

3.2 Multilevel memory management

Both of the books by Patterson & Hennessy have excellent discussions of memory hierarchies, with special attention paid to the design space for caches. Chapter 7 of reading 1.1.1 is very accessible; a substantially more intense version that is more performance-oriented and has additional speed-up tricks appears in the second book as chapter 5, reading 1.1.2 (see page 9-4). A lighter-weight treatment focused more on virtual memory, and including a discussion of stack algorithms, can be found in Chapter 3 of Tanenbaum's computer systems book, reading 1.2.2 (see page 9-6).

3.2.1 R[obert] A. Frieburghouse. Register allocation via usage counts. *Communications of the ACM 17,* 11 (November, 1974) pages 638-642.

This paper shows that compiler code generators must do multilevel memory management, and they have the same problems as do caches and paging systems.

3.2.2 R[ichard] L. Mattson, J. Gecsei, D[onald] R. Slutz, and I[rving] L. Traiger. Evaluation techniques for storage hierarchies. *IBM Systems Journal 9,* 2 (1970) pages 78-117.

The original reference on stack algorithms and their analysis, well written and in considerably more depth than the brief summaries that appear in modern textbooks.

3.2.3 Daniel L. Murphy. Storage organization and management in Tenex. *AFIPS Fall Joint Conference 41,* (December, 1972) pages 23-32.

This is a typical example of a time-sharing system that uses paging. It is probably easier to understand than some of the more recent mini- and micro-computer virtual memory systems that are muddied with tricks to stretch a too-small addressing architecture.

3.2.4 M[ahadev] Satyanarayanan and Dileep Bhandarkar. Design trade-offs in VAX-11 translation buffer organization. *Computer 14,* 12 (December, 1981) pages 103-111.

This paper nicely illustrates the range of issues encountered in designing a cache memory. As a bonus, it also provides an accessible brief summary of the VAX virtual memory addressing scheme.

3.2.5 Ted Kaehler. Virtual memory for an object-oriented language. *Byte 6,* 6 (August, 1981) pages 378-387.

See comment on reading 3.2.6, below.

3.2.6 Ted Kaehler and Glenn Krasner. LOOM: Large object-oriented memory for Smalltalk-80 systems. In Glenn Krasner, editor, *Smalltalk-80: Bits of History, Words of Advice.* Addison-Wesley, 1983. Pages 251-271. ISBN: 0-201-11669-3.

These two papers describe two generations of the memory-management system used in Smalltalk, an interactive programming system for desktop computers. A coherent virtual memory language support system provides for lots of small objects while taking into account address space allocation, multilevel memory management, and naming in an integrated way.

> The paper on the Woodstock File System, by Swinehart et al., reading 8.3.1 (see page 9-27), describes a file system that is organized as a multi-level memory management system.

4 Enforcing Modularity: Types, Clients, Servers, Kernels, and Threads

4.1 Type-extension as a system organizing tool

> The first system to use type-extension was the Cal time-sharing system, described in the paper by Lampson and Sturgis, reading 12.1.1 (see page 9-36). The second major run at organizing an operating system in an object-oriented style was Hydra, which had a full-scale type hierarchy; it is described in reading 12.2.1 (see page 9-36).

4.1.1 Michael D. Schroeder, David D. Clark, and Jerome H. Saltzer. The Multics kernel design project. *Sixth ACM Symposium on Operating Systems Principles,* in *Operating Systems Review 11,* 5 (November, 1977) pages 43-56.

This paper addresses a wide range of issues encountered in applying type-extension to Multics in order to simplify its internal organization. Many of these ideas were explored in even more depth in Philippe Janson's Ph.D. thesis, *Using type extension to organize virtual memory mechanisms,* M. I. T. Department of Electrical Engineering and Computer Science, August 1976. That thesis is also available as M.I.T. Laboratory for Computer Science Technical Report TR-167, September, 1976.

How to use data types and type managers in the large-scale operating system environment. Expanded version of the topic touched by reading 4.1.1.

4.2 Kernels and Remote Procedures

4.2.1 Per Brinch Hansen. The nucleus of a multiprogramming system. *Communications of the ACM 13,* 4 (April, 1970) pages 238-241.

The RC-4000 was the first, and may still be the best explained, system to use messages as the primary task coordination mechanism. It is also what would today be called a microkernel design.

4.2.2 Peter D. Varhol. Small kernels hit it big. *Byte 19,* 1 (January, 1994) pages 119-128.

A popularized treatment that indicates the latest trend in operating system design. Comparing this paper with the one by Brinch Hansen, reading 4.2.1, leads some readers to conclude that in 25 years nothing has changed but the buzzwords.

4.2.3 Andrew D. Birrell and Bruce Jay Nelson. Implementing remote procedure calls. *ACM Transactions on Computer Systems 2,* 1 (February, 1984) pages 39-59.

A well-written paper that shows first, the simplicity of the basic idea, second, the complexity required to deal with real implementations, and third, the refinements needed for high effectiveness.

4.2.4. Michael D. Schroeder and Michael Burrows. Performance of Firefly RPC. *ACM Transactions on Computer Systems 8,* 1 (February, 1990) pages 1-17. Originally published in Twelfth ACM Symposium on Operating Systems Principles, in *Operating Systems Review 23,* 5 (December, 1989) pages 102-113.

As a complement to the abstract discussion of remote procedure call in the previous paper, this one gives a concrete, blow-by-blow accounting of the steps required in a particular implementation, and then compares this accounting with overall time measurements. In addition to providing insight into the intrinsic costs of remote procedures, this work demonstrates that it is possible to do bottom-up performance analysis that correlates well with top-down measurements.

4.2.5 Brian N. Bershad, Thomas E. Anderson, Edward D. Lazowska, and Henry M. Levy. Lightweight remote procedure call. *ACM Transactions on Computer Systems 8,* 1 (February 1990) pages 37-55. Originally published in *Twelfth ACM Symposium on Operating Systems Principles,* in *Operating Systems Review 23,* 5 (December, 1989) pages 102-113.

4.2.6 Jochen Liedtke. Improving IPC by kernel design. *Fourteenth ACM Symposium on Operating Systems Principles,* in *Operating Systems Review 27,* 5 (December, 1993) pages 175-187.

These two papers develop techniques to allow local kernel-based client/server modularity to look just like remote client/server modularity to the application designer, while at the same time capturing the performance advantage that can come from being local.

4.3 Virtual Processors: Threads

4.3.1 Andrew D. Birrell. *An introduction to programming with threads.* Digital Equipment Corporation Systems Research Center Technical Report #35, January, 1989. 33 pages. (Appears also as chapter 4 of Greg Nelson, editor, *Systems Programming with Modula-3,* Prentice-Hall, 1991, pages 88-118.)

This is an excellent tutorial, explaining the fundamental issues clearly, and going on to show the subtleties involved in exploiting threads correctly and effectively.

4.3.2 Thomas E. Anderson, Brian N. Bershad, Edward D. Lazowska, and Henry M. Levy. Scheduler activations: effective kernel support for the user-level management of parallelism. *ACM Transactions on Computer Systems 10,* 1 (February, 1992) pages 53-79. Originally published in *Thirteenth ACM Symposium on Operating Systems Principles,* in *Operating Systems Review 25,* 5 (December, 1991) pages 95-109.

The distinction between user threads and kernel threads comes to the fore in this paper, which offers a way of getting the advantages of both by having the right kind of user/kernel thread interface. The paper also revisits the idea of a virtual machine, but in a multi-processor context.

4.3.3 David D. Clark. The structuring of systems using upcalls. *Tenth ACM Symposium on Operating Systems Principles*, in *Operating Systems Review 19*, 5 (December, 1985) pages 171-180.

Attempts to impose modular structure by strict layering sometimes manage to overlook the essence of what structure is actually appropriate. This paper describes a rather different inter-module organization that seems to be especially effective when dealing with network implementations.

4.3.4 Jerome H. Saltzer. *Traffic Control in a Multiplexed Computer System.* Ph.D. thesis, Massachusetts Institute of Technology, Department of Electrical Engineering, June, 1966. Also available as Project MAC Technical Report TR-30, 1966.

Describes what is probably the first systematic virtual processor design and implementation, the multi-processor multiplexing scheme used in the Multics system. Defines the coordination primitives *block* and *wakeup*, which are examples of binary semaphores assigned one per process.

5 Communications in Computer Systems

A good, in-depth tutorial introduction to this topic is Tanenbaum's book on networks, reading 1.2.3. An abbreviated version of that same material, sufficient for many readers, appears as chapter ten of his book on operating systems, reading 1.2.2 (see page 9-6). *Proceedings of the IEEE 66*, 11 (November, 1978), is a special issue of that journal devoted to packet switching, containing several papers mentioned under various topics here. Collectively they provide an extensive early bibliography on computer communications.

5.1 Networks

The book by Perlman on bridges and routers, reading 1.2.5 (see page 9-7), explains how the network layer really works.

5.1.1 David D. Clark, Kenneth T. Pogran, and David P. Reed. An introduction to local area networks. *Proceedings of the IEEE 66*, 11 (November, 1978) pages 1497-1517.

A basic tutorial on local area network communications. This paper characterizes the various modular components of a local area network, both interface and protocols, gives specific examples, and also explains how local area networks relate to larger, interconnected networks. The specific examples are by now out of date, but the rest of the material is timeless.

5.1.2 Robert M. Metcalfe and David R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM 19*, 7 (July, 1976) pages 395-404. Reprinted as chapter 26 of the book by Siewiorek, Bell, and Newell, reading 1.1.3 (see page 9-4).

This paper provides the design of what has proven to be the most popular local area network technology.

5.1.3 Michael D. Schroeder, Andrew D. Birrell, Michael Burrows, Hal Murray, Roger M. Needham, Thomas L. Rodeheffer, Edwin H. Satterthwaite, and Charles P. Thacker. Autonet: A high-speed, self-configuring local area network using point-to-point Links. *IEEE Journal on Selected Areas in Communications 9*, 8, (October 1991) pages 1318-1335.

Autonet is a mesh-connected local area network that uses an ATM-like technology, though with variable-length packets and without virtual channels; its goals of simple, field-changeable and automatic configuration resemble those of the Ethernet. A related paper, reading 10.2.3 (see page 9-32) by Rodeheffer and Schroeder, describes the automatic reconfiguration features of Autonet.

5.2 Protocols

5.2.1 Louis Pouzin and Hubert Zimmerman. A tutorial on protocols. *Proceedings of the IEEE 66,* 11 (November, 1978) pages 1346-1370.

This paper is well-written and provides perspective along with the details. Being written a long time ago turns out to be its major appeal. Because networks were not widely understood at the time, it was necessary to fully explain all of the assumptions and offer extensive analogies. This paper does an excellent job of both, and as a consequence it provides a useful complement to modern texts. One who is familiar with current network technology will frequently exclaim, “So that’s why the Internet works that way”, while reading this paper.

5.2.2 Vinton G. Cerf and Peter T. Kirstein. Issues in packet-network interconnection. *Proceedings of the IEEE 66,* 11 (November, 1978) pages 1386-1408.

At the time it was written, an emerging problem was interconnection of independently administered data communication networks. This paper explores the issues in both breadth and depth, a combination that more recent papers don’t provide.

5.2.3 David D. Clark. Modularity and efficiency in protocol implementation. *Internet Request for Comments RFC-817* (July, 1982). Available as Internet/World-Wide Web document <ftp://ds.internic.net/rfc/rfc817.txt>.

A discussion of the interaction between protocol design and protocol implementation, with special attention to the question of whether each protocol level requires a corresponding implementation level.

5.2.4 David D. Clark and David L. Tennenhouse. Architectural Considerations for a New Generation of Protocols. *ACM SIGCOMM ‘91 Conference: Communications Architectures and Protocols,* in *Computer Communication Review 20,* 4 (September, 1990) pages 200-208.

This paper captures 20 years of experience in protocol design and implementation and lays out the requirements the next few rounds of protocol design. The basic observation is that the performance requirements of future high-speed networks and applications will require that the layers used for protocol description not constrain implementations to be similarly layered. This paper is required reading for anyone who is developing a new protocol or protocol suite.

5.2.5 Danny Cohen. On holy wars and a plea for peace. *IEEE Computer 14,* 10 (October, 1981) pages 48-54.

An entertaining discussion of big-endian and little-endian arguments in protocol design.

5.2.6 Danny Cohen. Flow control for real-time communication. *Computer Communication Review 10,* 1-2, (January/April 1980) pages 41-47.

This brief item is the source of the “servant’s dilemma” a parable that provides helpful insight into why flow control decisions must involve the application.

> One popular protocol, Remote Procedure Call, is covered in depth in reading 4.2.3 (see page 9-14) by Birrell and Nelson, as well as section 10.3 of Tanenbaum’s Operating System book, reading 1.2.2 (see page 9-6).

5.3 Organization for communication

5.3.1 Leonard Kleinrock. Principles and lessons in packet communications. *Proceedings of the IEEE 66,* 11 (November, 1978) pages 1320-1329.

See comment on reading 5.3.2, below.

5.3.2 Lawrence G. Roberts. The evolution of packet switching. *Proceedings of the IEEE 66,* 11 (November, 1978) pages 1307-1313.

These two papers discuss experience with the ARPANET. Anyone faced with the need to design a network should, in addition to learning about current technology, look over these two papers, which focus on lessons learned and the sources of surprise.

5.3.3 J[erome] H. Saltzer, D[avid]. P. Reed, and D[avid]. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems 2*, 4 (November, 1984) pages 277-288. An earlier version appears in the *Second International Conference on Distributed Computing Systems,* (April, 1981) pages 504-512.

This paper proposes a design rationale for deciding which functions belong in which layers of a layered network implementation. One of the few papers available that provides a system design principle.

5.3.4 Leonard Kleinrock. The latency/bandwidth trade-off in gigabit networks. *IEEE Communications Magazine 30,* 4 (April, 1992) pages 36-40.

Technology is making gigabit/second data rates economically feasible over long distances. But long distances and high data rates conspire to change some fundamental properties of a packet network—latency becomes the dominant factor that limits applications. This paper provides a very good explanation of the problem.

5.3.5 Martin Reiser. Performance evaluation of data communication systems. *Proceedings of the IEEE 70*, 2 (February, 1982) pages 171-196.

A comprehensive survey of mathematical approaches to performance evaluation of networks: data link analysis, path control analysis, end-to-end analysis. Exceptional, though by now somewhat dated, bibliography.

5.4 Practical aspects

The starting place for network practice at the link level is with the books by McNamara, reading 1.2.6 (see page 9-7), and by Stallings, reading 1.2.7 (see page 9-7). At the network level, start with the book by Radia Perlman, reading 1.2.5 (see page 9-7). Or for the complete word on the Internet protocols, check out the following series of books.

5.4.1 W. Richard Stevens. *TCP/IP illustrated.* Addison Wesley; v. 1, 1994, ISBN 0-201-63346-9, 576 pages; v. 2 (with co-author Gary R. Wright,) 1995, ISBN 0-201-63354-x, 1174 pages.; v. 3, 1996, ISBN 0-201-63495-3, 328pages. *Volume 1: the protocols. Volume 2: The implementation. Volume 3: TCP for transactions, HTTP, NNTP, and the Unix domain protocols.*

These three volumes will tell you more than you wanted to know about how TCP/IP is implemented, using the network implementation of the Berkeley System Distribution for reference. The term “illustrated” refers more to computer printouts-listings of packet traces and programs-than to diagrams. If you want to know how some aspect of the Internet protocol suite is actually implemented, this is the place to look; unfortunately, it does not often explain why.

6 Naming and Binding in systems

Under the general topic of naming are three older papers that describe the abstract concepts more carefully than do most current treatments. Addressing architectures are usually well documented; this list includes only a few of the many papers in the area. File systems are often described as part of a larger system design, so many of the papers that would deserve to be mentioned here are cited under different topics.

Naming has emerged as a major complication when connecting computers with a data communications network, and several papers in the last group focus on that subject.

6.1. Naming

6.1.1 J[erome] H. Saltzer. Naming and binding of objects. Chapter 3.A 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. ISBN: 3-540-08755-9; 3-540-09812-7.

An in-depth tutorial on all aspects of naming in computer systems, with emphasis on the way the principles apply to addressing architectures and to file systems. Includes a substantial bibliography and a case study of naming in the Multics system.

6.1.2 Jack B. Dennis. Segmentation and the design of multiprogrammed computer systems. *Journal of the ACM 12*, 4 (October, 1965) pages 589-602.

The original paper outlining the advantages of providing naming support in a hardware architecture.

6.1.3 R[obert] S. Fabry. Capability-based addressing. *Communications of the ACM 17*, 7 (July, 1974) pages 403-412.

The first comprehensive treatment of capabilities, a mechanism introduced to provide protection but actually more of a naming feature.

6.2. Examples of addressing architectures

6.2.1 Elliott I. Organick. *The Multics System: an Examination of its Structure.* M.I.T. Press, Cambridge, Massachusetts, 1972. ISBN: 0-262-15012-3. 392 pages.

This book explores every detail and ramification of the extensive naming mechanisms of Multics, both in the addressing architecture and in the file system.

6.2.2 Elliott I. Organick. *Computer System Organization, The B5700/B6700 Series.* Academic Press, 1973. ISBN: 0-12-528250-8. 132 pages.

The Burroughs Descriptor system explained in this book is the only example of a hardware-supported naming system actually implemented before the advent of microprogramming.

6.2.3 George Radin and Peter R. Schneider. *An architecture for an extended machine with protected addressing.* IBM Poughkeepsie Laboratory Technical Report TR 00.2757, May, 1976.

Not widely circulated, this report describes (as best it can within confidentiality restrictions) an extremely ambitious naming architecture. Some of the ideas in this architecture later appeared in the IBM System/38 and AS/400 computer systems.

6.2.4 A[ndre] Bensoussan, C[harles] T. Clingen, and R[obert] C. Daley. The Multics virtual memory: concepts and design. *Communications of the ACM 15*, 5 (May, 1972) pages 308-318.

A good description of a system that pioneered the use of high-powered addressing architectures

See also the papers reprinted in the book by Siewiorek, Bell, and Newell, reading 1.1.3 (see page 9-4).

6.3 Examples of other naming systems

6.3.1 Bruce [G.] Lindsay. Object naming and catalog management for a distributed database manager. *Second International Conference on Distributed Computing Systems*, Paris, France, (April, 1981) pages 31-40. Also IBM San Jose Research Laboratory Technical Report RJ2914 (August, 1980). 17 pages.

A tutorial treatment of names as used in database systems. The paper begins with a better-than-average statement of requirements, then demonstrates how those requirements were met in the R* distributed database management system.

6.3.2 Derek C. Oppen, and Yogen K. Dalal. The Clearinghouse: A decentralized agent for locating named objects in a distributed environment. *Xerox Office Products Division Report OPD-T8103*, October, 1981. 52 pages.

A product version of some of the ideas pioneered in Grapevine, reading 12.11 (see page 9-39). This report provides design detail down to the level of subroutine interfaces.

6.3.3 Yogen K. Dalal and Robert S. Printis. 48-bit absolute Internet and Ethernet host numbers. *Seventh Data Communications Symposium*, Mexico City, Mexico, (October 1981) pages 240-245. Also Xerox Office Products Division Technical Report OPD-T8101 (July, 1981) 14 pages.

How hardware addresses are handled in the Ethernet local area network.

6.3.4 John Ioannidis, Dan Duchamp, and Gerald Q. Maguire, Jr. IP-based protocols for mobile internetworking. *ACM SIGCOMM '91 Conference: Communications Architectures and Protocols*, in *Computer Communication Review 21*, 4 (September, 1991) pages 235-245.

Addressing a laptop computer that is connected to a network by a radio link and that can move from place to place without disrupting network connections can be a challenge. This paper proposes implementing a virtual network and a scheme to continuously maintain a mapping of virtual network addresses to real ones.

6.3.5 Theodor Holm Nelson. *Literary Machines, Ed. 87.1*. Project Xanadu, San Antonio, Texas, 1987. ISBN 0-89347-056-2 (paperback). Various pagings.

Project Xanadu is an ambitious vision of a future in which books are replaced by information organized in the form of a naming network, in the form that today is called "hypertext". The book, being somewhat non-linear, is a primitive example of what Nelson advocates.

6.3.6 Butler W. Lampson. Designing a global name service. *Fifth ACM Conference on Principles of Distributed Computing* (August, 1986) pages 1-10.

A naming plan intended for large networks.

6.3.7 Apple Corporation, Caroline Rose, compilation editor. The alias manager. Chapter 27, pages 1-23 of *Inside Macintosh, Volume 6.* Addison-Wesley, 1991. ISBN 0-201-5775-0.

System 7 for the Macintosh has a remarkable mechanism for links between files, involving a mixture of hard links, soft links, and search heuristics. This mixture allows links to survive file and folder renaming, movement, and even deletion and restoration from backup. This chapter describes the mechanics in detail.

> Some of the most interesting examples of naming systems are found in case studies elsewhere in this reading list. For example, see the discussion of naming in Pilot, reading 11.2.3 (see page 9-34); in Grapevine, reading 12.11.1 (see page 9-39); the naming system of the Apollo Domain one-level-store, reading 12.7.3 (see page 9-38); and Murphy's paper about Tenex, reading 3.2.3 (see page 9-13).

7 Protection and Privacy in Computer Systems

7.1 Privacy

The fundamental book about privacy is reading 1.1.8 (see page 9-6) by Alan Westin.

7.1.1 Arthur R. Miller. *The Assault on Privacy*. University of Michigan Press, Ann Arbor, Michigan, 1971. ISBN: 0-47265500-0. 333 pages.

This book articulately spells out the potential effect of computerized data-gathering systems on privacy, and of possible approaches to improving legal protection. Part of the latter is now out of date because of advances in legislation, but most of this book is still of much interest.

7.1.2 Milton R. Wessel. *Freedom's Edge: The Computer Threat to Society*. Addison-Wesley, 1974. ISBN: 0-201-08543-7. 137 pages.

Written for a lay audience, this short paperback book thoroughly investigates the range of potential privacy-invading effects of the application of computer technology. The examples are badly out of date, but many of the issues raised in this book remain unresolved.

7.1.3 James Bamford. *The Puzzle Palace*. Houghton Mifflin, 1982. ISBN: 0-39531286-8. Also Penguin, 1983. ISBN: 0-14006748-5 (paperback). 465 pages.

A popularized but thorough peek inside some of the U.S. government's high-technology techniques for communications surveillance.

7.1.4 Rob Kling. Automated Welfare Client-Tracking and Service Integration: The Political Economy of Computing, *Communications of the ACM 21*, 6 (June, 1978) pages 484-493.

See comment on next paper.

7.1.5 Gene Dallaire. Computer matching: Should it be banned? John Shattuck. Computer matching is a serious threat to individual rights. Richard P. Kusserow. The government needs computer matching to root out waste and fraud. *Communications of the ACM 27* 6, (June, 1984) pages 537-545.

These two papers provide a carefully thought-out introduction to the consequences of technology on privacy. The second paper is actually a debate with pro and con positions on computer matching both well argued. All are reprinted in the book by Johnson and Nissenbaum, reading 1.3.11 (see page 9-9).

7.2. Encipherment as a Protection Mechanism

The fundamental book about cryptography applied to computer systems is reading 1.1.6 (see page 9-5), by Bruce Schneier. In the light of this book, the first few papers from the 1970's listed below are primarily of historical interest. An excellent brief introduction to cryptography in computer networks can be found in Tanenbaum's Computer Networks text, reading 1.2.3 (see page 9-6).There are also good treatments of cryptography in the books by Simson Garfinkel,reading 1.2.3 (see page 9-6) reading 1.2.8 (see page 9-7), and by Dorothy Denning, reading 1.3.9 (see page 9-9). Note that all of these books and papers focus on the application of cryptography, not on crypto-mathematics, which is a distinct area of specialization not covered in this reading list.

7.2.1 R[onald] L. Rivest, A[di] Shamir, and L[en] Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21,* 2 (February, 1978) pages 120-126.

First people out with a possibly workable public key system.

7.2.2 Data Encryption Standard. U. S. Department of Standards, National Bureau of Standards, Federal Information Processing Standard (FIPS) Publication #46, January, 1977 (#46-1 updated 1988; #46-2 updated 1994).

This is the official description of the national standard private key system, known as DES.

7.2.3 Whitfield Diffie and Martin E. Hellman. Exhaustive cryptanalysis of the NBS Data Encryption Standard. *Computer 10,* 6 (June, 1977) pages 74-84.

This is the unofficial analysis of how to break the DES, by building a special-purpose chip. Nearly two decades later, this proposal still seems to be the only promising attack on DES; the intervening improvements in hardware technology make the proposal look a little too feasible for comfort.

7.2.4 Horst Feistel, William A. Notz, and J. Lynn Smith. Some cryptographic techniques for machine-to-machine data communications. *Proceedings of the IEEE 63,* 11 (November, 1975) pages 1545-1554.

An older paper by the designers of the DES providing background on why it works the way it does. One should be aware that the design principles described in this paper are incomplete; the really significant design principles are classified as military secrets.

7.2.5 Li Gong, [T.] Mark A. Lomas, Roger M. Needham, and Jerome H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications 11,* 5 (June,1993) pages 648-656.

Introduces the idea of *verifiable plaintext,* material that someone who has guessed a key can use to verify that the key is correct. Verification is of concern whenever a password is used as an encryption key, because passwords are notoriously easy to guess. The paper points out that it is possible to design authentication protocols that do not contain verifiable plaintext, and it offers several examples.

7.2.6 [T.] Mark [A.] Lomas and Bruce Christianson. To whom am I speaking? Remote booting in a hostile world. *Computer 28,*1 (January, 1995), pages 50-54.

A short but very clearly-written paper that demonstrates the kind of thinking and analysis required to systematically construct a secure communication protocol. One unusual idea is the introduction of collision-rich hashing as an attack-detection mechanism.

7.2.7 Ross J. Anderson. Why cryptosystems fail. *Communications of the ACM 37,* 11 (November, 1994) pages 32-40.

A very nice analysis of what goes wrong in real-world cryptosystems—secure modules don’t necessary lead to secure systems—, and the applicability of systems thinking in their design. Anderson points out that merely doing the best possible design isn’t enough; a feedback loop that corrects errors in the design following experience in the field is equally important component that is sometimes forgotten.

7.3 Protection Architectures

7.3.1 Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE 63,* 9 (September, 1975) pages 1278-1308.

After twenty years, this paper still provides an effective treatment of protection mechanics in multi-user systems. Its emphasis on protection inside a single system, rather than between systems connected to a network is one of its chief shortcomings, along with antique examples and omission of newer techniques of certification such as authentication logic.

7.3.2 David D. Redell. *Naming and Protection in Extendible Operating Systems.* Ph.D. Thesis, University of California at Berkeley (September, 1974). Also available as M.I.T. Laboratory for Computer Science Technical Report TR-140, November, 1974. 161 pages.

The possibility that capabilities might be revocable was explored in depth in this thesis, using the idea that capabilities should contain indirect references to the objects to which they allow access. By changing or destroying the indirect entry, the owner of an object can render the capability useless, and not worry about whether or not it has been copied.

7.3.3 R[oger] M. Needham. Protection systems and protection implementations. *AFIPS Fall Joint Conference 41,* Part I (December, 1972,) pages 571-578.

This paper is probably as clear an explanation of capability systems as one is likely to find. There is another important paper on capabilities by Fabry, reading 6.1.3 (see page 9-20)

7.3.4 Michael D. Schroeder and Jerome H. Saltzer. A hardware architecture for implementing protection rings. *Communications of the ACM 15,* 3 (March, 1972), pages 157-170.

Protection rings are an easy-to-build generalization of the supervisor-user mode distinction, but they are of only limited utility. This paper explores the rationale for rings and describes some subtle details of design.

7.3.5 Lee M. Molho. Hardware aspects of secure computing. *AFIPS Spring Joint Computer Conference 36* (May, 1970,) pages 135-141.

The main item of interest in this paper is that it reports that delivered hardware does not necessarily match its wiring diagram, and that many failures in hardware protection features may be undetectable. Although the computer examined is now completely obsolete, modern hardware still seems to have the same problem and the lessons of this paper have not yet all been learned.

7.3.6 Frederick T. Grampp and Robert H. Morris. Unix operating system security. *Bell System Technical Journal 63,* 8, Part 2 (October, 1984) pages 1649-1672.

An unusually thoughtful and detailed discussion of lightweight security features, and both how they can be used effectively or misused badly. The chief defect in this paper is that the examples assume a thorough familiarity with Unix.

7.3.7 Robert [H.] Morris and Ken [L.] Thompson. Password security: A case history. *Communications of the ACM 22,* 11 (November, 1979) pages 594-597.

This paper is a model of how to explain something in an accessible way. With a minimum of jargon and an historical development designed to simplify things for the reader, it describes the Unix password security mechanism.

See also reading 1.3.8 (see page 9-9) by Garfinkel and Spafford, reading 12.1.1 (see page 9-36) by Lampson and Sturgis, and reading 12.5.2 (see page 9-37) by Saltzer.

7.4 Certification, Trusted Computer Systems and Security Kernels

7.4.1 Butler [W.] Lampson, Mart'n Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems 10,* 4 (November, 1992) pages 265-310.

One of a series of papers on a logic that can be used to reason systematically about authentication. This paper provides a relatively complete explication of the theory and shows how to apply it to the protocols of a distributed system.

7.4.2 Edward Wobber, Mart'n Abadi, Michael Burrows, and Butler W. Lampson. Authentication in the Taos operating system. *Fourteenth ACM Symposium on Operating Systems Principles,* in *Operating Systems Review 27,* 5 (December, 1993) pages 256-269.

This paper applies the authentication logic developed in reading 7.4.1 to an experimental operating system. In addition to providing a concrete example, the explanation of the authentication logic itself is a little more accessible than in the other paper.

7.4.3 Ken L. Thompson. Reflections on trusting trust. *Communications of the ACM 27,* 8 (August, 1984), pages 761-763.

Anyone who is seriously interested in developing trusted computer systems should think hard about the implications for verification that this paper raises. Thompson demonstrates the ease with which a compiler expert can insert undetectable Trojan Horses into a system.

7.4.4 Paul A. Karger, Mary Ellen Zurko, Douglas W. Bonin, Andrew H. Mason, and Clifford E. Kahn. A VMM security kernel for the VAX architecture. *1990 IEEE Computer Society Symposium on Security and Privacy* (May, 1990) pages 2-19.

In the 1970's, the U. S. Department of Defense undertook a research effort to create trusted computer systems for defense purposes, and in the process created quite a large body of literature on this subject. This paper distills most of the relevant ideas from that literature in a single, readable case study, and it also provides pointers to the key other papers if one wants to find out more about this set of ideas.

7.4.5 National Research Council. *Computers at Risk: Safe Computing in the Information Age.* National Academy Press, Washington D.C., 1991. ISBN 0-309-04388-3. 303 pages.

The National Academy of Sciences asked a committee of experienced computer system designers to assess, in the light of 20 years of research and development, what problems remained in protecting information in computers. The committee reported that most of the problems of 20 years earlier, though perhaps addressed in academic papers, still exist in practice, and actually have been made worse by the advent of computer networks.

7.4.6 Peter J. Denning, editor. *Computers Under Attack: Intruders, Worms, and Viruses.* Addison-Wesley, 1990 (Reprinted with corrections, 1991). ISBN 0-201-53062-8 (hardcover), 0-201-53067-8 (paperback). 567 pages.

In contrast with much of the shrill literature on the subject of its title, this volume collects a large number of thoughtful, and for the most part competent, published papers, comments, and other useful materials. The incident of the Internet Worm is thoroughly covered, and the treatment of viruses is also very thorough. The

last section of the book, on social, legal, and ethical implications contains quite a range of thought-provoking opinions.

7.4.7 David D. Clark and David. R. Wilson. A Comparison of Commercial and Military Computer Security Policies. *1987 IEEE Symposium on Security and Privacy* (April, 1987) pages 184-194.

This is a thought-provoking paper that outlines the requirements for security policy in commercial settings and argues that the lattice model is often not applicable. It suggests that these applications require a more object-oriented model in which data may be modified only by trusted programs.

> Almost half of the book by Dorothy Denning, reading 1.3.9 (see page 9-9), is on trusted computer systems. Chapter five of that book contains an overview of the lattice model of security policy, which is currently the basis for security policy of the military world and in consequence also the basis for most extant work on trusted computer systems.

8 Persistent Storage

8.1 Persistence per se

8.1.1 Jeff Rothenberg. Ensuring the longevity of digital documents. *Scientific American 272*, 1 (January, 1995) pages 42-47.

A superficial, but useful, overview of the problems of archiving digital information despite medium and technology obsolescence.

8.2 Properties of persistent storage devices

8.2.1 Chris Ruemmler and John Wilkes. An introduction to disk drive modeling. *Computer 27*, 3 (March, 1994) pages 17-28.

This paper is really two papers in one. The first five pages provide a wonderfully accessible explanation of how modern disk drives and controllers actually work. The rest of the paper, of interest primarily to performance modeling specialists, explores the problem of accurately simulating a complex disk drive, with measurement data to show the size of errors that arise from various modeling simplifications (or over-simplifications).

8.2.2 Randy H. Katz, Garth A. Gibson, and David A. Patterson. Disk system architectures for high performance computing. *Proceedings of the IEEE 77*, 12 (December, 1989) pages 1842-1857.

The reference paper on Redundant Arrays of Independent Disks (RAID). The first part reviews disk technology; the important stuff is the catalog of six varieties of RAID organization.

8.3 Examples of File Systems

> The CAP file system is one of the few that implements a genuine naming network. It is described in reading 12.4.2 (see page 9-37).

8.3.1 Daniel Swinehart, Gene McDaniel, and David [R.] Boggs. WFS: A simple shared file system for a distributed environment. *Seventh ACM Symposium on Operating Systems Principles,* in *Operating Systems Review 13*, 5 (December, 1979) pages 9-17.

Multilevel memory management with one of the levels remote, on another computer. This paper opens the door on the topic of distribution of function across connected cooperating computers. The authors had a specific goal of keeping things simple, so the relationship between mechanism and goal is much clearer than in more modern, but more elaborate, systems.

8.3.2 Marshall K. McKusick, William N. Joy, Samuel J. Leffler, and Robert S. Fabry. A fast file system for Unix. *ACM Transactions on Computer Systems 2*, 3 (August, 1984) pages 181-197.

The “fast file system” nicely demonstrates the trade-offs between performance and complexity in adding several well-known performance enhancement techniques, such as multiple block sizes and sector allocation based on adjacency, to a file system that was originally designed as the epitome of simplicity.

8.3.3 Mendel Rosenblum and John K. Ousterhout. The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems 10*, 1 (February, 1992) pages 26-52. Originally published in *Thirteenth ACM Symposium on Operating Systems Principles,* in *Operating Systems Review 25*, 5 (December, 1991) pages 1-15.

Although it has long been suggested that one could in principle store the contents of a file system on disk in the form of a finite log, this design is one of the few that demonstrate the full implications of that design strategy. The paper also presents a nice example of how to approach a system problem, by carefully defining the objective, measuring previous systems to get a benchmark, and then comparing performance as well as functional aspects that cannot be measured.

8.3.4 Wiebren de Jonge, M. Frans Kaashoek, and Wilson C. Hsieh. The logical disk: A new approach to improving file systems. *Fourteenth ACM Symposium on Operating Systems Principles,* in *Operating Systems Review 27*, 5 (December, 1993) pages 15-28.

This paper suggests a clever way to get much of the advantage of a log-structured file system at the disk level rather than at the file system level of the operating system.

8.3.5 James J. Kistler and M[ahadarev] Satyanarayanan. Disconnected operation in the Coda file system. *Thirteenth ACM Symposium on Operating Systems Principles,* in *Operating Systems Review 25*, 5 (December, 1991) pages 213-225.

Coda is a variation of the Andrew File System (AFS) that provides extra fault-tolerance features. It is notable for using the same underlying mechanism to deal both with accidental disconnection because of network partition and the intentional disconnection associated with portable computers. This paper is very well written.

See also reading 2.2.4 (see page 9-11), which describes the original Unix file system; reading 6.2.2 (see page 9-20), about the Burroughs file system; reading 12.1.1 (see page 9-36), about Cal and its file system, and reading 12.7.2 (see page 9-38), about the Apollo Domain file system.

9 Atomicity, Coordination, and Recovery

The best source on this topic is reading 1.1.5 (see page 9-5), the thousand-page book by Gray and Reuter, but it can be a bit overwhelming. For the most part, the papers in this section provide alternative views of the same material. The papers here by Gray can also be found as chapters in the book.

9.1 Atomicity

9.1.1 Jim [N.] Gray. The transaction concept: virtues and limitations. *Seventh International Conference on Very Large Data Bases*, (September, 1981) pages 144-154.

A very readable, masterful discussion of atomic transactions, with perspective on why some techniques work better than others. The “limitations” section identifies several good research problems. This paper reads so smoothly that it is deceptive: it explores some subtle and deep ideas that require contemplation to understand.

9.1.2 J[im] [N.] Gray. Notes on database operating systems. Chapter 3.F in Rudolf Bayer, et al., editors, *Operating Systems: An Advanced Course*, *Lecture Notes in Computer Science 60*, pages 393-481. Springer-Verlag, 1978, reprinted 1984. ISBN: 3-540-08755-9; 3-540-09812-7.

Describes a two-phase commit protocol for coordinating multiple updates with recovery. Oriented to database management, but ideas are fundamental.

9.1.3 David P. Reed. Implementing atomic actions on decentralized data. *ACM Transactions on Computer Systems 1*, 1 (February, 1983) pages 3-23.

Introduces the concepts of *possibilities* and *pseudo-time* as ways of achieving atomicity in distributed updates by labeling data versions systematically with time-stamps. Although the theoretical idea had been discussed for at least a decade, this is apparently the first practical proposal for using version histories as the basis for concurrency atomicity. For the full treatment, see the same author’s Ph.D. thesis, *Naming and Synchronization in a Decentralized Computer System*, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science (September, 1978). Also available as M.I.T. Laboratory for Computer Science Technical Report TR-205 (September, 1978). 181 pages.

9.1.4 Liba Svobodova. A reliable object-oriented data repository for a distributed computer system. *Eighth ACM Symposium on Operating Systems Principles*, in *Operating Systems Review 15*, 5 (December, 1981) pages 47-58.

A description of Swallow, an implementation of the system proposed in reading 9.1.3, above.

9.1.5 Warren A. Montgomery. *Robust Concurrency Control for a Distributed Information System.* Ph.D. thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, December, 1978. Also available as M.I.T. Laboratory for Computer Science Technical Report TR-207, January, 1979. 197 pages.

Describes alternative strategies that maximize parallel activity while achieving atomicity: maintaining multiple values for some variables, atomic broadcast of messages to achieve proper sequence.

9.1.6 D. B. Lomet. Process structuring, synchronization, and recovery using atomic actions. *Proceedings of an ACM Conference on Language Design for Reliable Software*, (March, 1977) pages 128-137. Published as *ACM SIGPLAN Notices 12*, 3 (March, 1977); *Operating Systems Review 11*, 2 (April, 1977); and *Software Engineering Notes 2*, 2 (March, 1977).

One of the first attempts to link atomicity with both recovery and coordination. Written from a language point of view, rather than an implementation perspective.

9.1.7 David K. Gifford. Weighted voting for replicated data. *Seventh ACM Symposium on Operating Systems Principles,* in *Operating Systems Review 13,* 5 (December, 1979) pages 150-162. Also available as Xerox Palo Alto Research Center Technical Report CSL-79-14 (September, 1979).

A replicated data algorithm that allows the trade-off between reliability and performance to be adjusted by assigning weights to each data copy and requiring transactions to collect a quorum of those weights before reading or writing.

9.1.8 Jim [N.] Gray, Paul McJones, Mike Blasgen, Bruce G. Lindsay, Raymond [A.] Lorie, Tom Price, Franco Putzolu, and Irving [L.] Traiger. The recovery manager of the System R database manager. *ACM Computing Surveys 13,* 2 (June, 1981) pages 223-242.

This paper is a case study of a sophisticated, real, high performance logging and locking system. It is one of the most interesting case studies of its type, because it shows the number of different, interacting mechanisms that are needed to construct a system that performs well.

See also the paper by Lampson and Sturgis, reading 13.2.8. (see page 9-43).

9.2 Synchronization and Deadlock

Every modern textbook covers these topics, probably more than their importance calls for. These readings extend the basic concepts in various directions.

9.2.1 E[dsger] W. Dijkstra. Co-operating sequential processes. In F. Genuys, editor, *Programming Languages,* NATO Advanced Study Institute, Villard-de-Lans, 1966. Academic Press, 1968. Pages 43-112.

Introduces semaphores, the synchronizing primitive most often used in academic exercises. Notable for its very careful, step-by-step development of the requirements for mutual exclusion and its implementation. Many modern treatments ignore the subtleties discussed here as if they were obvious; they aren’t, and if you want to really understand synchronization this is a paper you should read.

9.2.2 E[dsger] W. Dijkstra. Solution of a problem in concurrent programming control. *Communications of the ACM 8,* 9 (September, 1965) page 569.

This is the very brief paper in which Dijkstra first demonstrates that locks can be implemented entirely in software, relying on the hardware to guarantee only that read and write operations are relatively atomic.

9.2.3 David P. Reed and Rajendra K. Kanodia. Synchronization with eventcounts and sequencers. *Communications of the ACM 22,* 2 (February, 1979) pages 115-123.

An extremely simple coordination system that uses less powerful primitives for sequencing than for mutual exclusion; a consequence is simple correctness arguments.

9.2.4 Butler W. Lampson and David D. Redell. Experience with processes and monitors in Mesa. *Communications of the ACM 23,* 2 (February, 1980) pages 105-117.

A nice discussion of the pitfalls involved in integrating parallel activity coordination into a programming language.

9.2.5 Brian N. Bershad, David D. Redell, and John R. Ellis. Fast mutual exclusion for uniprocessors. *Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, (October, 1992) pages 223-233.

When threads are implemented on a single processor, the mechanics of mutual exclusion can be accomplished using idempotent instruction sequences which, if interrupted, can be restarted from the beginning. One can implement such sequences either by telling the operating system where they are located (explicit registration) or by restricting the setting of locks to clich s that an interrupt handler can recognize. This paper explores both of these ideas and measures their potential performance benefit.

> See also reading 4.2.1 (see page 9-14), by Brinch Hansen, which uses messages as a coordination technique, and reading 4.3.1 (see page 9-15), by Birrell, which describes a complete set of coordination primitives for programming with threads.

10 Limits and Reliability

10.1 Limits of current technology

10.1.1 John L. Hennessy and Norman P. Jouppi. Computer technology and architecture: An evolving interaction. *Computer 24*, 9, pages 18-29 (September, 1991).

10.1.2 Roger Wood. Magnetic megabits. *IEEE Spectrum 27*, 5 (May, 1990) pages 32-38.

A report on recent technical advances in magnetic recording technology. Note that survey papers like this one appear every four or five years in *IEEE Spectrum*; they generally describe the latest laboratory achievements, which will gradually appear in products over the ensuing five to ten years. *IEEE Computer* annually devotes its January issue to technology updates. Although they tend to be a little superficial, they are still handy for keeping up with the state of the art.

10.1.3 Richard F. Lyon. Cost, power, and parallelism in speech signal processing. Paper 15.1, *IEEE 1993 Custom Integrated Circuits Conference*, (May, 1993) pages 15.1.1-15.1.9.

One of the few readable papers that has so far appeared on a relatively new trade-off in system design: keeping the power consumption low in order to allow a system to be portable. Don't be thrown off by the term "speech signal processing," the issues explored here have general applicability.

10.2 Fault-Tolerance

> Chapter three of the book by Gray and Reuter, reading 1.1.5 (see page 9-5), provides a bedrock text on this subject. The book by Siewiorek and Swarz, reading 1.3.6 (see page 9-8) provides a somewhat broader discussion as well as a comprehensive set of case studies. Although fault-tolerance in mechanical systems has always been a concern, *ad hoc* techniques (when a system fails, patch it or its follow-on) have dominated. Unfortunately, rapid technology changes make this traditional technique ineffective for computer systems, so a small number of authors have endeavored to systematize things.

10.2.1 Jim [N.] Gray and Daniel P. Siewiorek. High-availability computer systems. *Computer 24*, 9 (September, 1991) pages 39-48.

A very nice, easy to read, overview of how high availability can be achieved.

10.2.2 Daniel P. Siewiorek. Architecture of fault-tolerant computers. *Computer 17*, 8 (August, 1984) pages 9-18.

This paper provides an excellent taxonomy, as well as a good overview of several architectural approaches to designing computers that continue running even when some single hardware component fails.

10.2.3 Thomas L. Rodeheffer and Michael D. Schroeder. Automatic reconfiguration in Autonet. *Thirteenth ACM Symposium on Operating Systems Principles,* in *Operating Systems Review 25,* 5 (December, 1991) pages 183-197. Also available as Digital Equipment Corporation Systems Research Center Technical Report #77, September, 1991, and as chapter eleven of Mullender's book, reading 1.3.4 (see page 9-8).

Describes some interesting *ad hoc*, but perhaps generalizable, techniques for achieving fault-tolerance in a multiply connected network. The idea of a skeptic, which takes the error rate into account in making decisions about what is working and what is not (a variant on the idea of systematic back-off) is probably the most interesting contribution of this paper.

> For an example of fault-tolerance in distributed file systems, see the paper on Coda by Kistler and Satyanaraynan, reading 8.3.5 (see page 9-28).

11 Control of Complexity

11.1 Wisdom about designing systems

> Before reading anything else on this topic, one should absorb the book by Brooks, *The Mythical Man-Month*, reading 1.1.7 (see page 9-5) and the essay by Simon, "The architecture of complexity," reading 2.1.3 (see page 9-10). After that, everything else is addendum.

11.1.1 Richard P. Gabriel. Worse is better. Excerpt from LISP: good news, bad news, how to win BIG, *AI Expert 6,* 6 (June, 1991) pages 33-35.

Explains why doing the thing expediently sometimes works out to be a better idea than doing the thing right.

11.1.2 Henry Petroski. Engineering: History and failure. *American Scientist 80,* 6 (November-December, 1992) pages 523-526.

Insight along the lines that one primary way that engineering makes progress is by making mistakes, studying them, and trying again. Petroski has visited this theme in two books, the most recent being reading 1.2.1 (see page 9-6).

11.1.3 Fernando J. Corbató. On building systems that will fail. *Communications of the ACM 34,* 9 (September, 1991) pages 72-81. (Reprinted in the book by Johnson and Nissenbaum, reading 1.3.11 (see page 9-9).)

The 1991 Turing Award Lecture. The idea here is that all ambitious systems will have failures, but those that were designed with that in mind are more likely to eventually succeed.

11.1.4 P[hillip] J. Plauger. Chocolate. *Embedded Systems Programming 7,* 3 (March, 1994) pages 81-84.

A remarkable insight, based on the observation that most failures in a bakery can be remedied by putting more chocolate into the mixture. The author manages, with only a modest stretch, to convert this observation into a more general technique of keeping recovery simple, so that it is likely to succeed.

11.1.5 Butler W. Lampson. Hints for computer system design. *Ninth ACM Symposium on Operating Systems Principles*, in *Operating Systems Review 17*, 5 (October, 1983) pages 33-48. Later republished, but with less satisfactory copy editing, in *IEEE Software 1*, 1 *(*January, 1984) pages 11-28.

An encapsulation of insights, expressed as principles that seem to apply to more than one case. Worth reading by all system designers.

11.1.6 James L. Woodward. What makes business programming hard? *BYTE* 7, 10 (October, 1982) pages 68-72.

This paper, by a bank vice-president who is responsible for his company's use of computing, provides quite a bit of insight into how it is that apparently simple real-world problems turn out to be very complex. The examples are quite easy to relate to.

11.1.7 Jon Bentley. The back of the envelope—programming pearls. *Communications of the ACM 27,* 3 (March, 1984) pages 180-184.

One of the most important tools of a system designer is the ability to make rough but quick estimates of how big, how long, how fast, or how expensive a design will be. This brief note extols the concept and gives several examples.

11.2 Keeping big systems under control: Case studies.

11.2.1 F[ernando] J. Corbató and C[harles] T. Clingen. A managerial view of the Multics system development. In Peter Wegner, *Research Directions in Software Technology*, M.I.T. Press, Cambridge, Massachusetts, 1979, pages 139-158. ISBN: 0-262-23096-8.

A large system case study emphasizing things that worked and things that didn't work to control development, with insight why.

11.2.2 W[illiam A.] Wulf, R[oy] Levin, and C. Pierson. Overview of the Hydra operating system development. *Fifth ACM Symposium on Operating Systems Principles,* in *Operating Systems Review 9,* 5 (November, 1975) pages 122-131.

A shorter case study of a smaller system, again with words of wisdom on what does and doesn't work in system implementation.

11.2.3 Thomas R. Horsley and William C. Lynch. Pilot: A software engineering case study. *Fourth International Conference on Software Engineering*, (September, 1979) pages 94-99.

These people designed a programming development system with just enough mechanism to do the job, and then used it to implement the Pilot operating system in the Mesa language at Xerox. This paper provides plenty of insight and wisdom about development systems.

11.2.4 Effy Oz. When Professional Standards are lax: the Confirm failure and its lessons. *Communications of the ACM 37,* 10 (October, 1994) pages 30-36.

Confirm is an airline/hotel/rental-car reservation system that never saw the light of day despite 4 years of work and an investment of more than $100M. It is one of many computer system developments that went out of control and finally were discarded without ever having been placed in service. One sees news reports of software disasters of similar magnitude about once each year. It is very difficult to get solid facts about system development failures, because no one wants to accept the blame, especially when lawsuits are pending. This paper suffers from a shortage of facts, and an over-simplistic recommendation that better ethics are all that are

needed to solve the problem (it seems likely that the ethics and management problems simply delayed recognition of the inevitable). Nevertheless, it provides a sobering view of how badly things can go wrong.

11.2.5 Nancy G. Leveson and Clark S. Turner. An investigation of the Therac-25 accidents. *Computer 26,* 7 (July, 1993) pages 18-41. (Reprinted in reading 1.3.11 (see page 9-9).)

Another sobering view of how badly things can go wrong. In this case, the software controller for a high-energy medical device was inadequately designed; the device was placed in service, and serious injuries ensued. This paper manages to inquire quite deeply into the source of the problems.

11.2.6. Philip M Boffey. Investigators agree N. Y. blackout of 1977 could have been avoided. *Science 201,* *** (15 September 1978) pages 994-996.

A fascinating description of how the electrical generation and distribution system of New York's Consolidated Edison fell apart when two supposedly tolerable faults occurred in close succession, recovery mechanisms did not work as expected, attempts to recover manually got bogged down by the system's complexity, and finally things cascaded out of control.

11.2.7. Joe Morgenstern. City perils: the fifty-nine-story crisis. *The New Yorker LXXI,* 14 (May 29, 1995) pages 45-53.

How an engineer responded to the realization that a skyscraper he had designed was in danger of collapsing in a hurricane.

11.3. Making your own contribution

One important technique for tackling complexity is to explain your system design to others. Before you set out to do that, however, it is advisable to look at some suggestions on how to write things in a way that others can understand.

11.3.1 George D. Gopen and Judith A. Swan. The science of scientific writing. *American Scientist 78,* 6 (November-December, 1990), pages 550-558.

Scientific writing isn't exactly the same as writing novels or writing for a newspaper. This paper dissects the situation and explains why.

11.3.2. Mary-Claire van Leunen. *A Handbook for Scholars.* Oxford University Press, Revised edition, 1992. ISBN 0-19-506953-6 (hardcover); 0-19-506954-4 (paperback). 348 pages.

This indispensable reference to scholarly writing not only tells and shows how to handle the mechanics of citations, quotations, footnotes, etc., but also, in the best tradition of Miss Manners, explains *why* you should do it in the recommended way.

11.3.3. Roy Levin and David D. Redell. An evaluation of the ninth SOSP submissions, or how (and how not) to write a good systems paper. *Operating Systems Review 17,* 3 (July, 1983) pages 35-40.

The co-chairs of the program committee for a symposium on systems reflected on the papers they had been required to review, and they set down their thoughts on what made some papers so much better (or worse) than others. Even if you don't intend to submit your paper to the next SOSP, you should read and heed the suggestions given here.

11.3.4 Bernard K. Forscher. Rules for referees. *Science 150, ***(October 15, 1965) pages 319-321.*

This note is nominally intended for referees, the people who evaluate submitted papers, but it includes a checklist of questions that every author should review before sending a paper in for publication. (It also contains a hilariously misguided comment about the future impact of technology on publication that illustrates the risks of getting off the topic on which you are an expert.)

11.3.5. Edward R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, Connecticut (P. O. Box 430, 06410), 1983. L/C 83-156861. 197 pages.

This privately-published book is undoubtedly the best explanation, with many splendid examples, of how to communicate numbers effectively.

11.3.6. Edward R. Tufte. *Envisioning Information.* Graphics Press, Cheshire, Connecticut (P. O. Box 430, 06410), 1990. L/C 90-166920. 126 pages.

The advanced version of the preceding reading. Carries the concepts beyond numbers, to the use of illustrations to communicate concepts.

12 Computer System Case Studies

The following collection of case study systems spans most of the range of interesting ideas extant in the computer system field today. All of the systems in this collection originated as advanced technology or research projects, and they range from widely used systems installed at thousands of sites, such as Unix and VM/370, through systems in modest commercial use with installations measured in tens, such as Multics and Tenex, to one-of-a-kind research systems (most of the remainder). There is no attempt to make this list comprehensive or all-encompassing; a deliberate choice has been made favoring systems that have interesting ideas, an exposition that offers insight, and preferably both. To be well-read in the computer systems area, one must have absorbed the ideas in a majority, if not all, of these case studies. For each system only the primary paper or papers are cited. As a general rule, the cited sources contain a bibliography of more detailed or specialized sources.

12.1 Cal

Cal appears to be the first system to make explicit use of types in the interface to the operating system. In addition to introducing this idea, the paper by Lampson and Sturgis is also very good at giving insight as to the pros and cons of various design decisions. Documented late, actually implemented in 1969.

12.1.1 Butler W. Lampson and Howard E. Sturgis. Reflections on an operating system design. *Communications of the ACM 19,* 5 (May, 1976) pages 251-265.

12.2 Hydra

The Hydra system explored in depth the possibilities of type extension as a system-organizing method from the ground up. There are several papers about Hydra, of which the following are the most important.

12.2.1 W[illiam A.] Wulf, E[lliot] Cohen, W. Corwin, A[nita] Jones, R[oy] Levin, C. Pierson, and F. Pollack. Hydra: The kernel of a multiprocessor operating system. *Communications of the ACM 17*, 6 (June, 1974) pages 337-345.

12.2.2 R[oy] Levin, E[lliot] Cohen, W. Corwin, F. Pollack, and W[illiam A.] Wulf. Policy/mechanism separation in Hydra. *Fifth ACM Symposium on Operating Systems Principles*, in *Operating Systems Review 9*, 5 (November, 1975) pages 132-140.

12.3 Unix

The prime reason for studying Unix is to see how shrewd design choices led (at least initially) to a very simple implementation of a very useful system. One should start with reading 2.2.4 (see page 9-11), by Ritchie and Thompson, which is the first of six papers in two special issues of the *Bell System Technical Journal* devoted to Unix and its applications; the other five are listed below. For a look at the much more complex Berkeley version of Unix, the book by Leffler et al., reading 1.3.7 (see page 9-9), provides a comprehensive guide to its entire design. A good, compact summary of the main features of Unix can be found in chapter 7 of Tanenbaum's operating systems book, reading 1.2.2 (see page 9-6). Listed under other topics are papers on the Unix Fast File System, reading 8.3.2 (see page 9-27) and on Unix security, reading 7.3.6 and reading 7.3.7 (see page 9-25).

12.3.1 Ken [L.] Thompson. Unix implementation. *Bell System Technical Journal 57*, 6, part 2, (1978) pages 1931-1946.

12.3.2 Dennis M. Ritchie. A retrospective. *Bell System Technical Journal 57*, 6, part 2, (1978) pages 1947-1969.

12.3.3 S[teven] R. Bourne. The Unix shell. *Bell System Technical Journal 57*, 6, part 2, (1978) pages 1971-1991.

12.3.4 Dennis M. Ritchie. The evolution of the Unix time-sharing system. *Bell System Technical Journal 63*, 8, part 2, (October, 1984) pages 1577-1593.

12.3.5 Rob Pike and Brian W. Kernighan. Program design in the Unix system environment. *Bell System Technical Journal 63*, 8, part 2, (October, 1984) pages 1595-1605.

12.4 The Cambridge Systems

The Computer Laboratory at the University of Cambridge is noted for taking slightly offbeat system ideas, shaking them out, showing how they can be useful, and gaining insight in the process. The CAP system is a capability protection system, with hardware support and a complete operating system—worked out further than any other capability system. And to complement this memory organization, its file system is organized as a naming network. The Cambridge Distributed Computing System extends the client-server model to banks of processors that are allocated as needed.

12.4.1 Maurice V. Wilkes and Roger M. Needham. *The Cambridge CAP Computer and its Operating System*. North Holland, 1979. ISBN: 0-444-00357-6 (hardcover); 0-444-00358-4 (paperback). 165 pages.

12.4.2 R[oger] M. Needham and A[ndrew] D. Birrell. The CAP filing system. *Sixth ACM Symposium on Operating Systems Principles*, in *Operating Systems Review 11,* 5 (November, 1977) pages 11-16.

12.4.3 Roger M. Needham and Andrew J. Herbert. *The Cambridge Distributed Computing System*. Addison-Wesley, 1982. ISBN: 0-201-14092-6 (paperback). 170 pages.

12.5 Multics

Multics is the original general-purpose computer utility. But more than that, it pioneered the integration of a lot of ideas into a single operating system: processes, a segmented one-level store, rings of protection, a hierarchical file system, access control lists, dynamic linking, programmable command language, stream I/O, replaceable interfaces, integrated performance measurement, and high-level-language implementation. All modern operating systems can trace many of their ideas to Multics; though some of those ideas have taken 20 years to show up again and users of Multics lament that other useful ideas have never reappeared.

12.5.1 F[ernando] J. Corbató, J[erome] H. Saltzer, and C[harles] T. Clingen. Multics—The first seven years. *AFIPS Spring Joint Conference 40*, (May, 1972,) pages 571-583.

12.5.2 Jerome H. Saltzer. Protection and control of information sharing in Multics. *Communications of the ACM 17,* 7 (July, 1974) pages 388-402.

12.5.3 Jerome H. Saltzer and John W. Gintell. The Instrumentation of Multics. *Communications of the ACM 13,* 8 (August, 1970) pages 495-500.

There are several other readings about Multics under various topics in this list. See the prospective paper by Corbató and Vyssotsky, reading 13.1.5; the paper on the Multics virtual memory by Bensoussan, et al., reading 6.2.4 (see page 9-20); the book on Multics by Organick (it concentrates mostly on the system support for the addressing hardware), reading 6.2.1 (see page 9-20); the thesis by Saltzer on processor multiplexing, reading 4.3.4 (see page 9-16); the paper by Freiburghouse on register allocation, reading 3.2.1 (see page 9-13); and the work by Schroeder et al., along with Janson's thesis on type-extension, reading 4.1.1 (see page 9-14). Thomas Van Vleck maintains an extensive archive of materials about Multics on the World-Wide Web at `<http://www.best.com/~thvv/multics.html>`; there are also several mirror sites.

12.6 3B20D and DMERT

The 3B20 is a highly-reliable Bell-System-designed computer that was for many years the basic engine of many of AT&T's telephone switching systems. The operating system, DMERT, contains Unix as a subset. This special issue of BSTJ contains sixteen papers, more than you wanted to know.

12.6.1 J. O. Becker, Editor. The 3B20D processor and DMERT operating system. *Bell System Technical Journal 62,* 1, part 2 (January, 1983) pages 167-428.

12.7 Apollo/Domain

The Apollo Domain system supports a different model for distributed function. It provides a shared primary memory called the Single Level Store, which extends transparently across the network. It is also one of the few systems to make substantial use of unstructured unique identifiers from a compact set as

object names. These papers don't just describe the design, they also discuss design trade-offs and the problems encountered in fleshing out these two ideas.

12.7.1 Paul J. Leach, Paul H. Levine, Bryan P. Douros, James A. Hamilton, David L. Nelson, and Bernard L. Stumpf. The architecture of an integrated local network. *IEEE Selected Areas of Communications SAC-1*, 5 (November, 1983) pages 842-857.

12.7.2 Paul J. Leach, Paul H. Levine, James A. Hamilton, and Bernard L. Stumpf. The file system of an integrated local network. *ACM Thirteenth Computer Science Conference* (March, 1985) pages 309-324.

12.7.3 Paul J. Leach, Bernard L. Stumpf, James A. Hamilton, and Paul H. Levine. UIDS as internal names in a distributed file system. In *ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, Ottawa, Ontario (August 18-20, 1982) pages 34-41.

12.8 Alto OS

The Alto OS is one of the first designs for a control program for personal computers. It thus pioneered several new design directions, including replaceability of anything.

12.8.1 Butler W. Lampson and Robert F. Sproull. An open operating system for a single-user machine. *Seventh ACM Symposium on Operating Systems Principles*, in *Operating Systems Review 13*, 5 (December, 1979) pages 98-105.

12.9 Pilot

Pilot was intended to be a production version of the Alto OS, with explicit support for higher level language, and a noticeable second system effect. In addition to the paper listed here, see also the one by Horsley and Lynch on the development system used for Pilot, reading 11.2.3 (see page 9-34).

12.9.1 David D. Redell, et al. Pilot: An operating system for a personal computer. *Communications of the ACM 23*, 2 (February, 1980) pages 81-92.

12.10 VM/370

The unusual feature of VM/370 is its creation of a strict, by-the-book, hardware virtual machine, thus providing the ability to run any system/370 program in a controlled environment.

12.10.1 Robert Jay Creasy. The origin of the VM/370 time-sharing system. *IBM Journal of Research and Development 25*, 5 (September, 1981) pages 483-490.

12.11 Grapevine

Grapevine is a service that maintains registries of user names for mail distribution and authentication. These papers provide early experience with a genuinely distributed database with attention to replication, fault-tolerance, message forwarding, security, and scaling to relatively large configurations. Reprints of the first two papers are included in the report containing the third one.

12.11.1 Andrew D. Birrell, Roy Levin, Roger M. Needham, and Michael D. Schroeder. Grapevine: An exercise in distributed computing. *Communications of the ACM 25,* 4 (April, 1982) pages 260-274. Originally presented at the *Eighth ACM Symposium on Operating Systems Principles.*

12.11.2 Michael D. Schroeder, Andrew D. Birrell, and Roger M. Needham. Experience with Grapevine: the growth of a distributed system. *ACM Trans. on Computer Systems 2,* 1 (February, 1984) pages 3-23. Originally presented at the *Ninth ACM Symposium on Operating Systems Principles.*

12.11.3 Andrew D. Birrell. The Grapevine interface. In A[ndrew] D. Birrell et al., *Grapevine: Two Papers and a Report,* Xerox Palo Alto Research Center Technical Report CSL-83-12, December, 1983. 56 pages.

12.12 Project Athena, Kerberos, and the X Window System

The Athena System is a network of engineering workstations and servers designed to support undergraduate education at M.I.T. The first small book provides a necessarily superficial, but still very useful, overview of Project Athena, which involved developing the Athena System, as well as deploying it and supporting a wide range of educational experiments on it. Kerberos is a working authentication and key distribution system for private (symmetric) encryption keys, based on the key distribution principles first publicly described in reading 13.2.5 (see page 9-42) by Needham and Schroeder. The X Window System, developed as part of Project Athena, is currently the window system of choice on practically every engineering workstation in the world. It provides good example of using the client/server model to achieve modularity. One of the main contributions of the X Window System is that it remedied a key defect that had crept into Unix when displays replaced typewriters: the display and keyboard were the only hardware-dependent parts of the Unix application programming interface. The X Window System allowed display-oriented Unix applications to be completely independent of the underlying hardware. In addition, the X Window System interposes an efficient network connection between the application and the display, allowing configuration flexibility in a distributed system.

12.12.1 George A. Champine. *M.I.T. Project Athena: A Model for Distributed Campus Computing.* Digital Press, Bedford, Massachusetts, 1991. ISBN 1-55558-072-6. 282 pages.

12.12.2 S[teven] P. Miller, B. C[lifford] Neuman, J[effrey] I. Schiller, and J[erome] H. Saltzer. Kerberos authentication and authorization system. Section E.2.1 of *Athena Technical Plan,* M. I. T. Project Athena, October 27, 1988.

12.12.3 Robert Scheifler and James Gettys. The X window system. *ACM Transactions on Graphics* 5, 2 (April 1986) pages 79-109.

12.13 The World-Wide Web

Many of the publications about the World-Wide Web are available only on the Web, and their citations are subject to change with time. However, there is one published starting reference that describes the general plan.

12.13.1 Tim Berners-Lee, Robert Cailiau, Ari Luotonen, Henrik Frystyk Nielsen, and Arthur Secret. The world-wide web. *Communications of the ACM 37,*8 (August, 1994) pages 76-82.

13. Papers of historic vision or perspective

Once in a while a paper comes along that either takes a sweeping new look at some aspect of systems design, or else has a dramatic vision of what future systems might do. The ideas in these papers often become part of

the standard baggage of all future writers in the area, but the reprises rarely do justice to the originals, which are worth reading if only to see how the mind of a visionary works. Here are some examples.

13.1. Dramatic visions

13.1.1. Vannevar Bush. As we may think. *Atlantic Monthly 176,* 1 (July, 1945) pages 101-108. Reprinted in Adele J. Goldberg, *A history of personal workstations,* Addison-Wesley, 1988, pages 237-247.

Bush looked at the (mostly analog) computers of 1945 and foresaw that they would someday be used as information engines to augment the human intellect.

13.1.2. John G. Kemeny, with comments by Robert M. Fano and Gilbert W. King. A library for 2000 a.d. In Martin Greenberger, editor, *Management and the Computer of the Future,* M. I. T. Press and John Wiley, 1962, pages 134-178.

It has taken 30 years for technology to advance far enough to make it possible to implement Kemeny's vision of how the library might evolve when computers are used in its support. Unfortunately, the engineering that is required still hasn't been done, so the vision has not yet been realized.

13.1.3. [Alan C. Kay, with the] Learning Research Group. *Personal Dynamic Media*. Xerox Palo Alto Research Center Systems Software Laboratory Technical Report SSL-76-1 (undated, circa March, 1976).

Alan Kay was imagining laptop computers and how they might be used long before most people had figured out that desktop computers might be a good idea. He gave many inspiring talks on the subject, but he rarely paused long enough to write anything down. Fortunately, his colleagues captured some of his thoughts in a technical report. An edited version of this report, with some pictures accidentally omitted, appeared in a journal the following year: Alan [C.] Kay and Adele Goldberg. Personal dynamic media. *IEEE Computer 10,* 3 (March, 1977) pages 31-41. This paper was reprinted with omitted pictures restored in Adele J. Goldberg, *A history of personal workstations,* Addison-Wesley, 1988, pages 254-263.

13.1.4 Doug[las] C. Engelbart. *Augmenting Human Intellect: A Conceptual Framework.* Research Report AFOSR-3223, Stanford Research Institute, Menlo Park, CA 94025,October, 1962. ***

Engelbart saw in the early 1960's that computer systems would someday be useful in a myriad of ways as personal tools. Unfortunately, the technology of his time, multi-million-dollar mainframes, was far too expensive to make his vision practical. Today's personal computers and engineering workstations have now incorporated many of his ideas.

13.1.5 F[ernando] J. Corbató and V[ictor] A. Vyssotsky. Introduction and overview of the Multics system. *AFIPS 1965 Fall Joint Computer Conference 27,* part I (1965), pages 185-196.

Working from a few, primitive examples of time-sharing systems, Corbató and his associates escalated the vision to an all-encompassing computer utility. Note that this paper is followed by five others (pages 185-247) about Multics.

13.2. Sweeping new looks

13.2.1. Jack B. Dennis and Earl C. Van Horn. Programming semantics for multiprogrammed computations. *Communications of the ACM 9,* 3 (March, 1966). Pages 143-155.

Set the ground rules for thinking about parallel activities, both the vocabulary and the semantics.

13.2.2. Claude E. Shannon. The communication theory of secrecy systems. *Bell System Technical Journal 28,* 4 (October, 1949) pages 656-715.***

The underpinnings of the theory of cryptography, in terms of information theory.

13.2.3 Whitfield Diffie and Martin E. Hellman. Privacy and authentication: an introduction to cryptography. *Proceedings of the IEEE 67,* 3 (March, 1979) pages 397-427.

The first really technically competent, paper on cryptography since Shannon in the unclassified literature; the paper that launched modern unclassified study. Includes a complete and scholarly bibliography.

13.2.4 Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory IT-22,* 6 (November, 1976) pages 644-654.

Diffie and Hellman invented public key cryptography; this is the paper in which they introduced the idea.

13.2.5 Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM 21,* 12 (December, 1978) pages 993-999.

The first accessible discussion of key distribution protocols, with applications, e.g., to electronic mail. The authors make an appeal for someone to develop a systematic way to check the validity of reasoning about encryption protocols, and indeed someone later found a minor flaw in one of their examples. The basic key-distribution idea discussed here was originally developed in a classified environment. A description that was so well veiled that the average reader would never notice it appeared in: Dennis K. Branstad. Security aspects of computer networks. *American Institute of Aeronautics and Astronautics Computer Network Systems Conference,* paper 73-427 (April, 1973).

13.2.6 J. S. Liptay. Structural aspects of the system/360 model 85: II. The cache. *IBM Systems Journal 7,* 1 (1968) pages 15-21.

The idea of a cache, look-aside, or slave memory had been suggested independently by Francis Lee and Maurice Wilkes some five years earlier, but it was not until the advent of LSI technology that it became feasible to actually build one in hardware. As a result, no one had seriously explored the design space options until the designers of the IBM System/360 model 85 had to come up with a real implementation. Once this paper appeared, a cache became a requirement for most later computer architectures.

13.2.7 Charles T. Davies., Jr. Data processing spheres of control. *IBM Systems Journal 17,* 2 (1978) pages 179-198. Charles T. Davies, Jr. Recovery semantics for a DB/DC system. *1973 ACM National Conference 28,* (August, 1973) pages 136-141.

A pair of papers that give a very high level discussion of “spheres of control”, a notion closely related to atomicity. Vague but thought-provoking. Everyone who writes about transactions mentions that they found these two papers inspiring.

13.2.8. Butler W. Lampson and Howard Sturgis. Crash recovery in a distributed data storage system. Working paper, Xerox Palo Alto Research Center, November, 1976, and April, 1979. (Never published)

Jim Gray calls the 1976 version “an underground classic.” The 1979 version has the first good definition of models of failure. Both describe algorithms for coordinating distributed updates; they are sufficiently different that both are worth reading.

13.2.9 Paul Baran, S. Boehm, and J. W. Smith. *On Distributed Communications.* A series of 11 memoranda of the RAND Corporation, Santa Monica, CA, August, 1964.***

Baran and his colleagues proposed the idea of packet switching, the basis for the modern Internet.

13.2.10 Lawrence G. Roberts and Barry D. Wessler. Computer network development to achieve resource sharing. *AFIPS Spring Joint Computer Conference 36*, (May, 1970,) pages 543-549.

This paper and four others in the same conference session (pages 543-597) are the first public description of the ARPANET, the first successful packet-switching network and the prototype for the Internet. Two years later, *AFIPS Spring Joint Computer Conference 40*, (1972) pages 243-298, contains five additional, closely related papers.

> Three other papers listed under specific topics also qualify as providing sweeping new looks or changing the way people that think about systems: Simon, The architecture of complexity, reading 2.1.3 (see page 9-10); Thompson, Reflections on trusting trust, reading 7.4.3 (see page 9-25); and Lampson, Hints for computer system design, reading 11.1.5 (see page 9-33)

Acknowledgement

The citations here have been accumulated, reviewed, and selected over more than twenty-five years, with the advice of many teaching staff members and students of M. I. T. subject 6.033, Engineering of Computer Systems. Particular thanks must go to Michael D. Schroeder, who uncovered several of the classic systems papers from places outside computer science where nobody else would have thought to look, Edward D. Lazowska, who maintains an extensive reading list at the University of Washington, Butler Lampson, for a thoughtful review of the list, and Frans Kaashoek, who has helped in a major updating in the springs of 1995 and 1996.

Appendix: Timely Readings Related to Systems

Spring, 1996, edition

Some of the most interesting materials about computer systems are “timely”, which means that they are of fleeting interest—they represent a current issue, the media has picked up on them, or they are continually in flux. This appendix provides a sampling of these items. Timely things are often found on-line these days, so some of these references are via World-Wide Web URL’s, delimited here by <pointed brackets>. Be warned that citations of on-line materials are fragile. The repositories that hold on-line materials do not generally operate by the archiving standards of libraries, so by the time you try to follow the citation, it may no longer be valid. Be warned also that this list contains media items, which, because of the pressure of deadlines, sometimes do not have the same depth and scholarly approach as do scientific journals. Nevertheless, they are useful for identifying significant issues.

1. Risks of Computer Systems.

1.1. General sources

1.1.1. Peter G. Neumann, moderator. *RISKS-Forum Digest.* Internet news group. <news:comp.risks> An archive <ftp://ftp.sri.com/risks/> is also available.

The RISKS forum is a long-standing, on-line moderated newsgroup with a publication rate of about once issue per week. Virtually anything that goes wrong in the world of computer systems gets reported here, and followed up with comments. At M. I. T. there is an Athena mailing list named risks that acts as a redistribution point, and there is also a partial archive in an Athena Discuss meeting.

1.1.2. Peter G. Neumann. *Computer-Related Risks.* Addison-Wesley, 1995. ISBN 0-201-55805-X (paperback).

A book by the moderator of the RISKS forum, which organizes and summarizes many of the things reported in that forum. Although the plan is good, the shortage of concrete facts on many incidents causes the book to read more like a laundry list than an expos ; it is more a reference than something to take to the beach to read.

1.2 Privacy sources

Quite a number of privacy resources are available on the Internet. The top-level resources are the four below; most other resources can be found via these.

1.2.1. Privacy Rights Clearinghouse. World-Wide Web site. <http://www.manymedia.com/prc/>

The Law School of the University of California at San Diego operates this clearinghouse, whose publications list includes a number of useful summaries of privacy-related matters. The Clearinghouse also (somewhat mysteriously) provides a Gopher service <gopher://pwa.acusd.edu:70/11/USDinfo/privacy> with different contents, including useful Fact Sheets and Issue Papers. One particularly notable item in the gopher site is a bibliography of privacy books and papers.

1.2.2. The Electronic Privacy Information Center (EPIC). World-Wide Web site. <http://www.digicash.com:80/epic/>

EPIC maintains an alerting system that watches for developments in Congress and elsewhere that might have an impact on privacy. EPIC has also collected an extensive bibliography of on-line and off-line resources on the subject of privacy at <http://cpsr.org/cpsr/privacy/epic/privacy_resources_faq.html>

1.2.3. Lauren Weinstein, moderator. *Computer Privacy Forum.* Distributed by e-mail. Also available in an archive. <http://www.vortex.com/privacy.htm>

This is a moderated mailing list that reports new and interesting incidents, legislation, and privacy concerns. The Privacy Forum is relatively new and sometimes duplicates material found in the RISKS forum. At M. I. T. you can subscribe to a local redistribution point by adding yourself to the Athena privacy-forum mailing list. Issues of this forum appear irregularly, typically every two or three weeks. Materials since May, 1993 are also available as an Athena Discuss meeting.

1.2.4. The Computer Privacy Digest. Internet news group. <news:comp.society.privacy> An archive is also available. <gopher://gopher.cs.uwm.edu/>

This is another moderated newsgroup. Contributions to this newsgroup sometimes overlap or duplicate those of the Computer Privacy Forum, though the two moderators have distinctly different approaches. Someone at the University of Wisconsin in Madison maintains the gopher archive of this digest.

1.3. Discussions about privacy.

1.3.1 We know you're reading this. *The Economist 338,* 7952 (February 10, 1996) pages 27-28.

Brief discussion of the contemporary privacy scene in the United States; tracking. logging, information mining. A sidebar on page 28 discusses the 1996 extension of the 1909 Comstock Act to make it illegal to transmit information about abortion on the Internet.

1.3.2 Chris Hibbert. Social security number FAQ: What to do when they ask for your social security number. Reposted frequently to an Internet news group. <news:comp.society.privacy> The current version is maintained in an Internet FAQ archive. <ftp://rtfm.mit.edu/pub/usenet/comp.society.privacy>.

One of the better overviews of the situation surrounding social security numbers, which are often used as a kind of unique identifier. This description makes it clear what the hazards are.

1.3.3 Larry Tye. No Private Lives. Series of four leader articles with six accompanying sidebars, *The Boston Globe 244,* 67, 68, 69, and 70 (September 5, 6, 7, and 8, 1993). The articles and sidebars carry the following headlines:

Privacy lost in high-tech era. (September 5) pages 1, 18, and 19.
Hidden assets are an open book to Fla. firm. (September 5) sidebar, page 18.
When the cost of information is too high. (September 5) sidebar, page 19
List-makers draw a bead on many. (September 6) pages 1, 12, and 13.
(With Maria Van Schuyver.) Technology tests privacy in the workplace. (September 6) sidebar, page 13
German system puts a lid on data. (September 7) pages 1, 10, and 11.
EC may force new look at privacy. (September 7) sidebar, page 10.
Britons find some comfort under cameras' gaze. (September 7) sidebar, page 11.
Proposed ÔBill of Rights' would limit personal data. (September 8) page 1, 12, and 13.

As press accesses more data, limits weighed. (September 8) sidebar, page 13.

Overall, a very good look, from a media perspective, at the current stand-off between technology and privacy.

1.4. Wiretapping—Clipper and the Digital Telephony Initiative

There is a major argument developing between American law-enforcement officials and privacy advocates on the appropriate response to the conversion of telephony from analog to digital form. End-to-end digital communications can easily be encrypted, so wire-tapping is not as easy as it was in the analog world. Clipper is a proposed standard encryption chip with the feature that a copy of its decrypting key is held by the government. The Digital Telephony Initiative is a proposed requirement that telephone companies develop and install features that allow routing copies of particular digital streams to law enforcement agencies.

1.4.1. Lance J. Hoffman, editor. *Building in Big Brother.* Springer-Verlag, 1995. ISBN: 0-367-94441-9 (paperback). 560 pages.

Relentlessly broad in its coverage, this book reprints practically everything of significance that has appeared so far on the concept of encryption key escrow, digital wiretapping, and encryption export. Its primary virtue is that it saves the need to track down things that have been published in many diverse places. Its primary defect is that the many sources overlap, so the amount of redundancy involved in reading it cover to cover is frustrating. The opening chapters, which are intended to provide an introduction to encryption for less scientific readers, are somewhat oversimplified, but no punches are pulled in the original papers, press releases, congressional testimony, and position papers that follow. Although the dimensions of the debate described here are timeless, the details will become outdated quite rapidly. Nevertheless, for the moment it makes an excellent reference.

2. Disasters.

2.1 Software development projects that went awry

2.1.1 Ralph T. King, Jr. California DMV's computer overhaul ends up as costly ride to junk heap. *Wall Street Journal,* East coast edition, April 27, 1994, page B7.***

This report is light on facts, but one thing seems clear: the California Department of Motor Vehicles spent $44M and the system had to be junked.

2.1.2 Tom Davis. Software Usability II. Internal memorandum of the Silicon Graphics Corporation (October 5, 1993). Posted by Jerry Leichter under the title "Stress analysis of a software project" in *RISKS-Forum Digest 15,* 80 (April 28, 1994).

An unusually candid memorandum, probably not intended for public consumption, about the problems of controlling development of large, complex software systems. The next item is a follow-up.

2.1.3 Tom Davis (posted by Joan Eslinger). Re: Stress analysis of a software project. *RISKS-Forum Digest 15,* 81 (April 29, 1994).

The good news appears in this follow-up, which reports that the previous memorandum succeeded in getting management attention, the management process changed dramatically, and the system in question was delivered and works.

2.1.4. Gary Stix. Aging Airways. *Scientific American 270,* 5 (May, 1994) pages 96-104.

The U. S. Federal Aviation Administration seems to be having some trouble modernizing the systems that underlie the U. S. Air Route Control System.

2.2. Other disasters

2.2.1. Eugene H. Spafford. Crisis and aftermath. *Communications of the ACM* 32, 6 (June, 1989) pages 678-687.

In early November, 1988, many people came to work one day to discover that their Internet-connected Unix systems were in the grip of an automated intruder that, having penetrated their system, was exploring it to find clues about other systems it could also penetrate. This event shook up quite a number of people, and the ensuing blizzard of media coverage was notable more for heat than light. This paper is one of the few factual and more reasoned discussions of the incident.

2.2.2. Brian Hayes. Waiting for 01-01-00. *American Scientist 83,* 1 (January-February, 1995) pages 12-15.

This paper explores the debate as to whether there may be a disaster lying in wait with the coming turn of century. The concern is that many computer programs store and manipulate calendar dates using only the last two digits of the year, and the turnover from 99 to 00 may reveal so many bugs in systems that we depend on that society will come to a screeching halt.

3. Other impacts of computer system technology.

3.0.1 Eli M. Noam. Electronics and the dim future of the university. *Science 270,* 5234 (October 13, 1995) pages 247-249.

This paper suggests that the computer and communications revolution will imperil the university by undermining all three of its fundamental reasons for existing. Whether he is right or wrong, the paper brings up a lot of interesting things to discuss about the impact of high-technology systems on society.

3.0.2 Robert W. Lucky. Reflections: Riding the wave. *IEEE Spectrum 30,* 7 (July, 1993) page 19.

How choosing the right technology wave is a matter of judgement, not entirely technical.

3.0.3 Barry Cipra. Electronic time-stamping: The notary public goes digital. *Science 261,* 5118 (July 9, 1993) pages 162-163. Includes sidebar: All the hash that's fit to print.

An interesting application for one-way encryption: establishing the integrity of a document by publishing its signature.

3.0.4. Sue Conger and Karen D. Loch, editors. Ethics and Computer Use. *Communications of the ACM 38,* 12 (December, 1995) pages 30-384.

This special section of an issue of the *CACM* contains six papers, ranging from a tutorial about ethics for people who have never had a chance to pursue the topic in formal study, to case studies of ethical issues that have arisen in computer systems design and use. Saves the need to read all of the papers in the book by Johnson & Nissenbaum.

3.0.5 Gary Stix. The speed of write. *Scientific American 270,* 12 (December, 1994) pages 106-111.

The high-energy physics research community has adopted whole-heartedly the distribution of preprints of research papers, undermining traditional journal publishing systems and raising the question of where it will all end.

3.0.6 [Steven R. Lerman and] Jerome H. Saltzer. Teaching students about responsible use of computers. *Communications of the ACM* 32, 6 (June, 1989) page 704. The publication omits the name of the first author.

This is the first official M.I.T. statement of principles of responsible use of computing and networks. Note that there is an omitted line that makes complete hash of one principle.

3.0.7 Larry Press. Before the Altair: The history of personal computing. *Communications of the ACM 36*, 9 (September, 1993) pages 27-33.

3.1. Surveys from The Economist.

3.1.1. The software revolution—A survey of defence technology. *The Economist 335*, 7919 (June 10, 1995) 24 pages, follows page 50.

3.1.2. The accidental superhighway—A survey of the Internet. *The Economist 336*, 7921 (July 1, 1995) 22 pages, follows page 50.

3.1.3. The death of distance—A survey of telecommunications. *The Economist 336*, 7934 (September 30, 1995) 32 pages, follows page 64.

The weekly news magazine *The Economist* publishes occasionally special survey inserts, of which two or three each year relate to computer and communications systems. The quality of reporting and editing on such technical topics is remarkable, especially for a general-circulation publication. The above are three recent examples.

4. The World-Wide Web on the World-Wide Web.

These four sources provide technical details on the machinery that underlies the World-Wide Web.

4.0.1 Daniel W. Connolly and Tim Berners-Lee. Names and Addresses, URIs, URLs, URNs, URCs. `<http://www.w3.org/hypertext/WWW/Addressing/Addressing.html>` (January 3, 1996).

4.0.2 Tim Berners-Lee, Larry Masinter, and M. McCahill. Uniform Resource Locators (URL). Internet RFC-1738 (December, 1994). `<http://www.w3.org/hypertext/WWW/Addressing/rfc1738.txt>`

4.0.3 Unknown. Hypertext Transfer Protocol (HTTP). `<http://www.w3.org/hypertext/WWW/Protocols/>`.

4.0.4 Daniel W. Connolly. HyperText Markup Language (HTML). `<http://www.w3.org/hypertext/WWW/MarkUp/>`.

5. Hot technology.

Every year some new technology areas suddenly appear to be the wave of the future. Initially, it is hard to decide both whether or not they really are, and also whether the first papers are really the right ones to read. Some of these items may make their way into the longer-term recommended reading list after a year or two of experience.

5.0.1 Dawson R. Engler, M. Frans Kaashoek, and James O'Toole, Jr. Exokernel: an operating system architecture for application-level resource management. *Fifteenth ACM Symposium on Operating Systems Principles*, in *Operating Systems Review 29*, 5 (December, 1995) pages 251-266.

Exokernel takes the idea of a microkernel to its logical extreme, exposing everything possible to higher-level applications. By using a capability-like revocation scheme that is also visible at the higher level it can withdraw resources gracefully or rudely, as the situation requires. Because of the potential for high performance, this set of ideas may well begin to make its way into the next generation of operating systems.

5.0.2. James Gosling. Java intermediate Bytecodes. *ACM SIGPLAN Workshop on Intermediate Representations*, in *SIGPLAN Notices 30*, 3 (March 1995) pages 111-118. (see also `<http://java.sun.com/>`)

Java is a language that Jim Gosling has been working on for several years; in its latest incarnation it is showing up as a language to write programs that World-Wide Web servers can download into your Web browser for execution there. By requiring that the stack have typed entries, it allegedly becomes possible for a verifier in the client to establish that it is safe to compile and run the program. This paper, on first blush, is not terribly accessible; it appears that the reader is expected to have had ten years of experience in compiler code generator design. But if one glosses over the heavy-duty details the higher-level concepts stand out and aren't that hard to extract.

5.0.3. Mark Weiser, Brent Welch, Alan Demers and Scott Shenker. Scheduling for reduced CPU energy. *First USENIX Symposium on Operating Systems Design and Implementation* (November, 1994) pages 13-23.

A design dimension that has become important, especially in the design of portable equipment, is power consumption. This very accessible paper describes one aspect of the associated trade-offs using intuitively satisfying models. (Unfortunately, the charts and figures accompanying the section that presents simulation results are a model of how *not* to display quantitative information.)

Computer science engineering projects involve designing and development of various application-based software. Computer science project topics can be implemented by a number of tools such as Java, .NET, Oracle, etc. The list of computer science project ideas is as follows. Computer Science Project Topics for Engineering Students. CSE Projects for Engineering Students. Voice based E-mail for the Blind.Â Hello sir,Iâ€™m 2nd year student.I need model topics which should be based on computer science.And in the model should have uniqueness and the model should not be mini. please give me model topics. Reply. Tarun Agarwal says