Biographies

Thomas Haigh, Editor
University of Pennsylvania

Biography

Per Brinch Hansen

Per Brinch Hansen was born 13 November 1938, in Copenhagen, Denmark. He is a pioneer in the development of operating system principles and parallel programming languages.

Significant work

Written over a period of more than 30 years, Brinch Hansen's technical papers and textbooks describe a relentless search for simplicity exemplified by the RC 4000 multiprogramming system; the Solo operating system; the monitor notation for modular parallel programming; the parallel programming languages Concurrent Pascal, Edison, Joyce, and SuperPascal; and his scientific programs for parallel architectures. Throughout his career, Brinch Hansen attempted to recognize the essence of complex software problems and express them in terms of a small number of abstract concepts. The technical ideas he recognized are now standard material in textbooks on operating systems and parallel programming. His scientific work can be divided into three phases: operating system principles (1966–1972), parallel programming languages (1972–1988), and computational science (1988–1998). In recent years, he has devoted increasing energy toward documenting the historical origins of modern operating systems and concurrent programming concepts.

Brinch Hansen studied electrical engineering at the Technical University of Denmark. After graduating in 1963 he joined the Danish computer company Regnecentralen. There he was a member of a Cobol compiler group headed by Peter Naur and Jorn Jensen. He discovered that writing is a rigorous test of simplicity: "It is just not possible to write convincingly about ideas that cannot be understood."[1] At Regnecentralen he was responsible for the architecture and software of the RC 4000 minicomputer. The RC 4000 real-time control system was his first experience with multiprogramming and semaphores (1967). The project succeeded because they used the simplest possible techniques to solve an unfamiliar problem of modest size.

The RC 4000 multiprogramming system introduced the novel idea of a system kernel for parallel processes and message communication that can be extended with a variety of operating systems.[2] Brinch Hansen left the computer industry in 1970 and became a university researcher in the US.

The implementation techniques of operating systems were reasonably well understood in the late sixties. But most systems were too large and poorly described to be studied in detail. While Edsger Dijkstra had clarified fundamental aspects of process synchronization, most of the literature on operating systems emphasized implementation details of particular systems rather than general concepts. The terminology was unsystematic and incomplete.

At the time there really were no suitable textbooks on operating systems. Universities were therefore unable to teach core courses on the subject. In 1970 Alan Perlis invited Brinch Hansen to spend a year at Carnegie Mellon University where he wrote the first comprehensive textbook on the principles of operating systems, *Operating System Principles*.[3]

Of the process of understanding operating systems he said:

> While writing the book I reached the conclusion that operating systems are not radically different from other programs. They are just large programs based on the principles of a more fundamental subject: parallel programming.
>
> Starting from a concise definition of the purpose of an operating system, I divided the subject into five major areas. First, I presented the principles of parallel programming as the essence of operating systems. Then I described processor management, memory management, scheduling algorithms, and resource protection as techniques for implementing parallel processes.
>
> I defined operating system concepts by algorithms written in Pascal extended with an (unimplemented) notation for structured multiprogramming. The book includes a concise vocabulary of operating system terminology, which is used consistently throughout the text.[3]

The combined work of Dijkstra, Tony Hoare, and Brinch Hansen on programming notation for operating system concepts led to the initial development of parallel programming languages. It began in 1971 with Hoare's notation for a conditional critical region that delays one or more processes until a Boolean expression is true.

Unfortunately, this elegant idea is inefficient because it requires periodic reevaluation of the expression as long as it is false. One of Brinch Hansen's first papers on programming language concepts proposed an (equally inefficient) variant of conditional critical regions for priority scheduling. A more original idea was his introduction of scheduling queues, which eliminated superfluous evaluation (but complicated the programming somewhat).[4]

Dijkstra, Hoare, and Brinch Hansen suggested another parallel programming concept in 1971: the monitor, which combines synchronization procedures with the shared variables upon which they operate. Brinch Hansen's operating system book[3] introduced a programming notation for monitors (shared classes), based on the class concept of Simula 67. Somewhat later, Hoare published a similar notation for monitors. His proposal included a variant of Brinch Hansen's scheduling queues ("conditions").

By the fall of 1972 Brinch Hansen was already committed to the goal of developing a parallel programming language with a modular ("object-oriented") notation for processes and monitors. At the California Institute of Technology he defined the programming language Concurrent Pascal, which supports monitors and parallel processes for implementation of modular operating systems. Because synchronization errors can be extremely difficult to locate by program testing, Concurrent Pascal was designed to permit the detection of such errors during compilation. With the help of a couple of students, a portable implementation of the language was running on a PDP-11 minicomputer at the end of 1974. Concurrent Pascal had obvious limitations by today's standards. But, in 1975, it laid the foundation for the development of programming languages with abstract concepts for parallelism.

Brinch Hansen used Concurrent Pascal to program the portable operating system Solo as a realistic test of the new programming language. The most significant contribution of Solo was undoubtedly that the program text was short enough to be published in its entirety in a computer journal.[5] The portable implementation of Concurrent Pascal and Solo was highly successful in spreading the software to almost 200 installations worldwide. The time was now ripe for a book about the principles of abstract parallel programming, and so he produced *The Architecture of Concurrent Programs*[6] while at the University of Southern California. The book includes the complete text of three model operating systems written in Concurrent Pascal.

Technology was now moving from multiprocessors with shared memory toward multicomputers with distributed memory. For microcomputer networks, Brinch Hansen proposed a combination of processes and monitors called distributed processes, which communicate by means of synchronized "remote procedure calls."[7]

After Concurrent Pascal, Brinch Hansen designed the parallel programming languages, Edison and Joyce, to experiment with Hoare's concepts of conditional critical regions and synchronous message communication, respectively.[8,9]

At the end of the preface to his collection of classic papers,[1] Brinch Hansen said of his activities:

> In the 1990s the programming problems of operating systems have surfaced again in parallel scientific computing: there is a serious need for machine-independent programming languages and algorithms. To understand this challenge I spent five years writing portable multicomputer algorithms for typical problems in science and engineering. My book, *Studies in Computational Science*,[10] describes this work.

In 2002, the IEEE Computer Society awarded Brinch Hansen its IEEE Computer Pioneer Award for his accomplishments in operating systems and concurrent programming. Fittingly, this recognition came after several years of work on his part to document historical developments in these areas, and to celebrate and illuminate the accomplishments of his fellow pioneers. His efforts in this area have already produced two edited volumes, one focused on classic operating systems[11] and the other on the major breakthroughs in concurrent programming.[12] Brinch Hansen recently joined the editorial board of the *IEEE Annals of the History of Computing*, where we hope to benefit from his expertise and enthusiasm.

Quotations

"Writing is a rigorous test of simplicity: It is just not possible to write convincingly about ideas that cannot be understood."[1]

"Programming is the art of writing essays in crystal clear prose and making them executable."[6]

"Although the practical demands of software design make simplicity essential, a more profound reason is to be found in the nature of creative work. The joy of discovery and the pleasure of making something work are the

Background of Per Brinch Hansen

Education: Brinch Hansen earned an MS in electrical engineering in 1963 from Technical University of Denmark. **Professional experience:** Regnecentralen (Copenhagen): 1963–1970, systems programmer; 1967–1970, head of software development; Carnegie Mellon University: 1970–1972, research associate; California Institute of Technology: 1972–1976, associate professor of computer science; University of Southern California: 1976–1982, professor; 1982–1984, Henry Salvatori Professor of Computer Science; Technical University of Denmark: 1984–1987, professor; Syracuse University: 1987–present, distinguished professor of computer science. **Honors and awards:** Doctor Technices, Technical University of Denmark, 1978; Fellow, IEEE, 1985; Chancellor's Medal, Syracuse University, 1989; IEEE Computer Pioneer Award, 2002.

Deaths Noted

Recent months have seen the passing of many important figures in the world of computing. While obituaries of the following men are already widely available, the Editorial Board hopes to publish more considered biographies and remembrances in future issues of *Annals*. If you have stories related to them, or feel able to write a balanced and authoritative overview of one these lives, then please contact Thomas Haigh, editor of the Biographies department, at thaigh@sas.upenn.edu.

- *Saul Amarel* (*1928-2002*). Founder of the Rutgers computer science department and an influential researcher in artificial intelligence.

- *Samuel D. Conte* (*1917-2002*). Founder of America's first computer science degree program at Purdue University.

- *Ole-Johan Dahl* (*1931-2002*) *and Kristen Nygaard* (*1926-2002*). Developers of the Simula I modeling language and Simula 67 general-purpose language, generally recognized as the ancestors of all subsequent object-oriented programming languages. They were the winners of the 2001 Turing Award, the highest honor of the ACM.

- *Edsger Wybe Dijkstra* (*1930-2002*). Developer of key operating systems concepts, pioneer in numerous areas of computer science, and elegant polemicist for mathematic rigor in all aspects of programming. Winner of the 1972 Turing Award and numerous other honors.

- *Keith Uncapher* (*1922-2002*). Former director of the Computer Science Division of the RAND Corporation and founder of the Information Sciences Institute at the University of Southern California, in which he oversaw seminal work in packet-switching technology and the Internet's domain name system.

most powerful drives in science and engineering. To sustain this motivation, a software engineer must look for astonishing simplicities and beautiful patterns of design."[13]

Biographical bibliography

P. Brinch Hansen, "The Programmer as a Young Dog," *The Search for Simplicity: Essays in Parallel Programming,* IEEE CS Press, 1996, pp. 142-146.

P. Brinch Hansen, "Monitors and Concurrent Pascal: A Personal History," *History of Programming Languages II*, T.J. Bergin Jr. and R.G. Gibson Jr., eds., ACM Press, 1996, pp. 121-172.

Selected publications

In addition to those listed in the "References and notes" section, here are some of Brinch Hansen's significant publications:

P. Brinch Hansen, "The Programming Language Concurrent Pascal," *IEEE Trans. Software Engineering*, vol. 1, no. 2, June 1975, pp. 199-207.

P. Brinch Hansen, "SuperPascal—A Publication Language for Parallel Scientific Computing," *Concurrency-Practice and Experience*, vol. 6, no. 5, Aug. 1994, pp. 461-483.

P. Brinch Hansen, *Programming for Everyone in Java*, Springer-Verlag, 1999.

References and notes

1. P. Brinch Hansen, *The Search for Simplicity: Essays in Parallel Programming*, IEEE CS Press, 1996.
2. P. Brinch Hansen, "The Nucleus of a Multiprogramming System," *Comm. ACM*, vol. 13, no. 4, Apr. 1970, pp. 238-242.
3. P. Brinch Hansen, *Operating System Principles*, Prentice-Hall, 1973.
4. P. Brinch Hansen, "Structured Multiprogramming," *Comm. ACM*, vol. 15, no. 7, July 1972, pp. 574-578.
5. P. Brinch Hansen, "The Solo Operating System," *Software-Practice and Experience*, vol. 6, no. 2, Apr.–June 1976, pp. 141-205.
6. P. Brinch Hansen, *The Architecture of Concurrent Programs*, Prentice-Hall, 1977.
7. P. Brinch Hansen, "Distributed Processes: A Concurrent Programming Concept," *Comm. ACM*, vol. 21, no. 11, Nov. 1978, pp. 934-941.
8. P. Brinch Hansen, "The Design of Edison," *Software-Practice and Experience*, vol. 11, no. 4, Apr. 1981, pp. 363-396.
9. P. Brinch Hansen, "Joyce—A Programming Language for Distributed Systems," *Software-Practice and Experience*, vol. 17, no. 1, Jan. 1987, pp. 29-50.
10. P. Brinch Hansen, *Studies in Computational Science: Parallel Programming Paradigms*, Prentice Hall, 1995.
11. P. Brinch Hansen, ed., *Classic Operating Systems: From Batch Processing to Distributed Systems*, Springer-Verlag, 2001.
12. P. Brinch Hansen, ed., *The Origin of Concurrent Programming: From Semaphores to Remote Procedure Calls*, Springer-Verlag, 2002.

13. P. Brinch Hansen, *Programming a Personal Computer*, Prentice Hall, 1983.

J.A.N. Lee
janlee@cs.vt.edu

Thomas Haigh
University of Pennsylvania
thaigh@sas.upenn.edu

Obituary

Edward Louis Glaser

Edward Louis Glaser, 1986. (Courtesy Cheryl Glaser.)

Edward Louis Glaser was born 7 October 1929; he died in October 1991 in Los Angeles. Blind since childhood, he once quipped "I've been accused of having superb far vision, but my near vision isn't too good."[1] Glaser was an early visionary of computer science. He served in industry, led government research projects, and helped to found an academic computer science department. His insights into the nature of computation illustrated how little this new discipline has in common with the world of the senses.

Losing sight, gaining confidence

Glaser was born with severely impaired vision and was completely blind by the age of eight. His parents, James and Margaret Glaser, had difficulty accepting their son's "disability." He found solace in music and spent one year convincing his apprehensive parents to let him play the piano. Once they consented, he showed extraordinary talent and made rapid progress. By his early teens, he was often sneaking out to improvise with professional musicians at bars near his hometown of Glencoe, Illinois.

When not entertaining crowds with his keyboard wizardry, Glaser often spent time developing his sleight of hand. A family friend had introduced Glaser to magic shortly after blindness overtook the boy, and Glaser was immediately hooked. Performing magic tricks in front of other people increased his confidence, encouraging him to act in several Gilbert and Sullivan operettas while attending North Shore Country Day School.

Glaser established a reputation as a "gadget freak" at an early age, devoting hours to the design of circuits at age 11.[2] At 14, he earned a ham radio operator's license and spent much of his time constructing a radio. Upon completion, he freely explored the airwaves, on equal footing with the sighted.[3] He yearned for similar independence in the outside world. That became a possibility two years later when Glaser received a seeing-eye dog. His new friend provided an autonomy that "opened up a whole new life for him. It was the first time he hadn't had to rely on another person."[4] Over his lifetime, he would grow attached to six such dogs, all of which served as his eyes and his constant companions.

With new poise, Glaser prepared to tackle Dartmouth College as a music and drama major. However, in the middle of his sophomore year, he discovered mathematics. The transition from music to a more technical field felt natural, Glaser explained in a later interview: "Whether you are writing music or designing computers, the creative intellect becomes strengthened."[3] Though he exchanged drama for physics, Glaser kept his performance ability fresh by hosting an early morning radio show on the college station.

Significant work

Graduating Phi Beta Kappa from Dartmouth with a degree in physics, Glaser initially wanted to pursue employment in actuarial mathematics. He faced rejection from insurance companies that refused to believe a blind man could operate their IBM machines. One company was willing to hire Glaser, but its personnel manager told him "that handicapped people couldn't do work as well as anybody else [so] they were going to hire him at 25 percent under the minimum."[5] Glaser turned that job down.

Months later, Glaser visited his fiancée at Bryn Mawr during a campus showcase of opportunities for women in the sciences. Having previously memorized a few boards and their wiring, Glaser approached the men from IBM. He later said,

> So I went up to see the SSEC [Selective Sequence Electronic Calculator], and I figured as long as I was here, I could ask some questions about this IBM stuff that I couldn't do. They talked to me for a few hours and figured out there was nothing I couldn't do, so I got hired. I was the only one they got from Bryn Mawr that year.[6]

Glaser excelled at IBM, aiding in the design of the IBM 604, 650, 701, and Wooden Wheel