USENIX notes

USENIX MEMBER BENEFITS

Members of the USENIX Association receive the following benefits:

FREE SUBSCRIPTION to *;login:*, the Association's magazine, published six times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on such topics as security, Perl, VoIP, and operating systems, book reviews, and summaries of sessions at USENIX conferences.

ACCESS TO ;LOGIN: online from October 1997 to this month: www.usenix.org/publications/login/.

ACCESS TO PAPERS from USENIX conferences online: www.usenix.org/publications/library/proceedings/

THE RIGHT TO VOTE on matters affecting the Association, its bylaws, and election of its directors and officers.

DISCOUNTS on registration fees for all USENIX conferences.

DISCOUNTS on the purchase of proceedings and CD-ROMs from USENIX conferences.

SPECIAL DISCOUNTS on a variety of products, books, software, and periodicals. For details, see www.usenix.org/membership/specialdisc.html.

TO JOIN SAGE, see www.usenix.org/membership/classes.html#sage.

FOR MORE INFORMATION regarding membership or benefits, please see www.usenix.org/membership/ or contact office@usenix.org. Phone: 510-528-8649

USENIX BOARD OF DIRECTORS

Communicate directly with the USENIX Board of Directors by writing to *board@usenix.org*.

PRESIDENT

Michael B. Jones, *mike@usenix.org*

VICE PRESIDENT

Clem Cole, *clem@usenix.org*

SECRETARY

Alva Couch, *alva@usenix.org*

TREASURER

Theodore Ts'o, *ted@usenix.org*

DIRECTORS

Matt Blaze, *matt@usenix.org*

Rémy Evard, *remy@usenix.org*

Niels Provos, *niels@usenix.org*

Margo Seltzer, *margo@usenix.org*

EXECUTIVE DIRECTOR

Ellie Young, *ellie@usenix.org*

NOTICE OF ANNUAL MEETING

The USENIX Association's Annual Meeting with the membership and the Board of Directors will be held during the 2007 USENIX Annual Technical Conference, June 17–22, 2007, Santa Clara, California. The time and place of the meeting will be announced on-site and on the conference Web site, www.usenix.org/usenix07.

IN MEMORIAM: JOHN W. BACKUS, 1924–2007

Alex Aiken
Professor of Computer Science, Stanford University

Computing has been such an integral part of everyday life for so long that many people alive today don't remember a time when it wasn't so. In the 1950s, digital computers were only a few years old, and very little of what we take for granted about modern computers had been invented. One thing was already apparent, however: the new, powerful hardware needed equally powerful software if it was to be of use to anyone, and software was turning out to be surprisingly difficult, time-consuming, and expensive to write.

A big part of the problem was that software at the time was written directly in the machine's native assembly language. As anyone who has ever written in assembly knows, it takes a lot of error-prone coding to get anything done that way.

John Backus, who was working at IBM at the time, had the idea that programming could be greatly improved if the programmer could write more abstract and less machine-specific commands and if the program could be organized in a way that seemed more natural for humans. A special program called a compiler would take care of translating

from the higher-level language into the machine's language.

The obvious problem with this idea, a difficulty that led some to predict it would fail, was the concern that the translation would end up being much less efficient than a carefully written assembly program; since machines were very expensive, computer time was a valuable commodity.

But John was literally a visionary; he had a vision of what could be. He led a team that spent three years (1954–1957) designing and implementing the programming language and its compiler.

Within a few months of its first release, FORTRAN (for FORmula TRANslation) was a huge success for IBM, dramatically improving software productivity. FORTRAN was an even greater intellectual triumph, for it changed forever the way people thought about programming computers and led directly to a surge of interest in the design and implementation of high-level computer languages.

John was also a prototype of the modern computer science research manager, an oxymoron if there ever was one. John always claimed that his contribution to the FORTRAN project was only breaking up the chess games after lunch, that really he hadn't done anything himself.

His characteristic modesty aside, his management style was one of almost no management at all, taking only the role of articulating the vision of where the project should go, both to inspire those working with him and to convince his backers to continue funding the project.

The key was to let the project be guided by the technical constraints of the problem they wanted to solve, which was poorly understood at the time, and not by a manager's early guess at what the solution should be. It's a great trick, but really quite difficult to pull off, since projects of FORTRAN's ambition succeed only after many failures. It requires both great technical ability and a gift for connecting with people to keep an effort like that on track, but John made it look easy.

After FORTRAN, John went on to invent BNF (or Backus-Naur Form), the notation used universally today to describe the syntax of programming languages. He also was a major force behind the early development of functional languages, championing even higher-level programming as necessary to move beyond languages of the FORTRAN era.

For his work he won the Turing Award and the Draper Prize, among many other awards, but he never stopped being a researcher, always interested in new ideas and ways to make programming easier and more useful.