Prehistory, from an operating system point of view, was 1950. On the earliest digital computers, each user's job ran separately and had exclusive use of the computer for the duration of the program. The program communicated with I/O devices and managed other resources without assistance. When the program completed, an operator halted the computer and manually prepared the next job. The mid-1950s saw the dawn of batch processing and early operating systems which did little more than load programs and manage I/O devices. More general-purpose systems did not emerge until the mid-1960s. Many of these were multimode systems. They not only provided batch processing and early operating systems which did little more than load programs and manage I/O devices. More general-purpose systems did not emerge until the mid-1960s. Many of these were multimode systems. They not only provided batch processing, but also included modes for time sharing and some real-time processing. These behemoths tried, within a single system, to provide every functionality to all users.

Twenty-year history

Operating system refinement and consolidation came in the 1970s, when important concepts were developed, disappeared for a time, and then reappeared. For example, virtual memory, first shown on the Atlas system in 1959, did not become available in commercial systems until 1972, when IBM released it as part of its general product line.

For a time, many postulated that the operating system might disappear and that the user had no need to know of the central services provided. The Apple Macintosh was cited as an example. It has no command line interpreter and no "system call" interface. The environment is simply a collection of library routines that provide access to the windowing system. But isn't this an operating system too?

Today, operating systems provide security, networking, distributed access, graphics, time-sharing, multiprocess, multithreading, and multiprocessor capabilities but, unlike their ancestors, are not intended to be all things to all people within a single system. Like the computers on which they run, operating systems vary in size, complexity, and functionality to fulfill users' requirements. Some, like Digital Research's CP/M and Microsoft's MS-DOS, are little more than program loaders that also support a local disk drive. Others, such as DEC's VMS, Data General's AOS/VS, and AT&T's Unix, provide support for, and simultaneous access to, a large number of different peripheral devices. Sophisticated environments have been developed under these operating systems.

Dramatic changes in operating systems have resulted from enhancements to the hardware technology, experience with existing systems, development of new applications and needs, and, of course, imagination. Ten years ago, networking was an interesting concept with a few, unique interconnected systems; today, it's become a requirement. The combined developments of Ethernet and commodity microprocessors have provided a new environment of high-performance, single-user graphical workstations connected over local area networks. LANs filled with personal workstations have created the demand for efficient use of computing and storage resources. Wide area networks, with the potential for worldwide connectivity, provide the possibility of global access to computing resources. Unfortunately, global agreement over the protocols used to access these resources seems unlikely, thereby placing new demands on operating systems to provide users with uniform and integrated data access.

Multiprocessor architectures have moved from experimental research labs into the mainstream of the computing world. The operating system, as the software foundation for these systems, is being asked to efficiently meet new requirements. Modern operating system implementations need to address such issues as load balancing across processors, gang scheduling, and algorithm locking.

Special-issue articles

Progress in the 20 short years of operating system history has been remarkable. The field has grown with the new, modern computer architectures. The operating system provides an ever-more-useful view of the system. This special issue of Computer looks at just a few of the recent developments. Its goal is to show where operating systems research has led us and where it might lead us in the future.
Early computer users placed the computer in the center of their world, with one or more large computers serving the needs of many. These systems were maintained by a dedicated staff. The advent of workstations shifted the focus to the user. The resulting problem was that users could no longer share data and now had the responsibility of administering their own systems—a task they were not trained to do. While no one wanted to eliminate workstations, it was recognized that a balance had to be struck. To accomplish this, we use a few large systems as file and computation servers and leave the user interface on the workstation. The first article in this issue, "Scalable, Secure, and Highly Available Distributed File Access," by Mahadev Satyanarayanan, discusses the evolution of a distributed file system that provides highly available access to data through a combination of read/write replication and data caching.

Networking has become commonplace. What has not become common is the use of a single communications protocol. In some circumstances, this would be useful, but different protocols provide different functionality and capabilities. Most workstations provide only a single protocol family. Access to multiple protocol families, so that users can access different networks, will be an important requirement for the future. In the second article, "The x-Kernel: A Platform for Accessing Internet Resources," Larry Peterson, Norman Hutchinson, Sean O'Malley, and Herman Rao discuss an operating system design to facilitate the implementation of multiple network protocols.

It has always amazed us that, after 20 years of wrestling with multiprocess operating systems, we are still trying to determine how to best schedule multiple processes. One reason is that the tasks we ask our systems to perform constantly change. Another is that computer architectures have also changed. Scheduling on a multiprocessor system should be different from scheduling on a uniprocessor— if not, we have not achieved optimal use of this new architecture. David L. Black's article, "Scheduling Support for Concurrency and Parallelism in the Mach Operating System," discusses scheduling techniques and policies for multiprocessor systems.

Advances in networking and workstations have not been independent activities. A natural outgrowth of these two developments has been an effort to make networking, multiple computer systems appear as a single entity. A new operating system that implements this approach is Amoeba. Its design and implementation are discussed in "Amoeba: A Distributed Operating System for the 1990s," by Sape J. Mullender, Guido van Rossum, Andrew S. Tanenbaum, Robbert van Renesse, and Hans van Staveren.

These first four articles discuss the results of software development efforts. "Algorithms Implementing Distributed Shared Memory," by Michael Stumm and Songnian Zhou, mathematically analyzes the trade-offs of several distributed shared memory algorithms.

Multiprocessor systems have gained commercial success in the past few years. These systems, like all computers, continue to evolve. The final article, "Distributed Hierarchical Control for Parallel Processing," by Urr Peetelson and Larry Rudolph, discusses a new and unique approach to multiprocessor that allows dynamic repartitioning.

Acknowledgments

We would like to thank the many authors who submitted articles to this special issue. The quantity and quality of those submissions was more than we expected. We would also like to thank the many referees who donated countless hours to reading and commenting on papers. It is through their effort that issues such as this are possible.

We would also like to thank our colleagues and the management at Encore Computer. Editing this issue has taken a great deal of time—our management has been more patient with us than we had any right to expect. Our colleagues were called upon more than once to referee papers during the initial review period and to read and reread the final manuscripts.

Finally, we would like to thank Bruce Shriver, editor-in-chief of Computer, for his time, assistance, and suggestions, which enhanced the quality of this issue.

References


Joseph Boykin is the manager of Mach Operating System Development with Encore Computer Corporation. The Mach OS group is responsible for the transition of Mach from a research prototype to a commercial product, as well as the use of Mach within research projects funded from sources both within and outside of Encore.

The Open Software Foundation will use Encore's implementation of Mach as the basis for the OSF/1 operating system. Encore's Mach OS group will provide consulting, design, and implementation support to OSF.

Boykin has held several leadership positions within the IEEE Computer Society. Currently, he is the society's treasurer and an Executive Committee member. He has chaired the Technical Committee on Operating Systems and the Workshop on Workstation Operating Systems.

Boykin holds both an MS in computer science and an MA in psychology. His graduate work was done at Ohio State and Pennsylvania State Universities.

Susan J. LoVerso has been a software engineer at Encore Computer Corporation since 1987 as a member of the Mach Operating System Development Group. Currently, she is working on parallelizing the 4.4BSD file system for the OSF/1 operating system.

LoVerso received her master's degree in computer science from the State University of New York at Buffalo. As a member of the Education Committee of the Women's Initiative for Technology Leadership in New England, she often speaks at local high schools about careers in the sciences and mathematics.

Readers can write to the guest editors at Encore Computer, 257 Cedar Hill St., Marlborough, MA 01752-3089. Their e-mail addresses are boykin@encore.com and sue@encore.com.