Early Timesharing

Multiprograming, multitasking, and timesharing

Mainframe computers were expensive. The central computer that Dartmouth bought in 1975 cost more than four million dollars, a great deal of money for a small university. Moreover, these computers had huge economies of scale. Several years later, when Cornell studied the options for replacing its mainframe computer, an IBM 370/65 cost twice as much as a smaller IBM 370/55, yet was four times as powerful. With such economies of scale, every organization followed the same strategy: buy the largest computer you could afford and find ways to maximize the amount of work done on it.

Computer hardware improved rapidly during the 1960s, both in performance and reliability, but processor cycles were a scarce resource not to be wasted. Computers of the previous generation had run one program at a time and the central processor might spend much of its time idling. For example, a program to process a tray of punched cards would spend most of its time waiting for the next card to be read.

By the middle of the decade computers were powerful enough to run more than one program at a time. The first technique that allowed several tasks to be run simultaneously was called multiprogramming. Each program was given a priority. At any given moment the operating system would offer the central processor to the highest priority program. Suppose that the highest priority program copied punched cards on to tape. The program would start to read a card and then wait while the card was read. This caused an interrupt and the scheduler would start up the next program in priority. This might send data to a line printer. This program too would spend much of its time waiting and the central processor could be assigned to another program. Rather unintuitively, the highest priority was given to the program that used the central processor least. This primitive form of multiprogramming required the operators to understand the characteristics of each program. It evolved into multitasking, where the user specifies the characteristics of a job, such as memory requirements, and the operating system schedules the execution. Modern operating system use preemptive multitasking, where the operating system manages the entire scheduling, interrupting (or preempting) tasks after a predetermined time interval.

IBM's OS/360, released in 1966, is often considered to be the first modern operating system. Its focus was on efficient batch processing for commercial data processing. The painful lessons learned during its development were the subject of the first book on software engineering, *The Mythical Man Month* by Fred Brooks.

The beginnings of timesharing

Batch processing operating systems, with their emphasis on long-running data processing jobs, were unsuitable for academic computing. Universities, therefore, developed timesharing to give their faculty and students direct access to a central computer. The early systems included Multics at MIT, Titan at Cambridge University, and the Dartmouth Time Sharing System (DTSS). MIT and Cambridge emphasized the needs of researchers, while Dartmouth's focus was on undergraduate education. My first glimpse at timesharing was in the summer of 1965 when I visited Dartmouth and sat in on a lecture by John Kemeny on Basic programming. A year later, when my wife was a

graduate student at MIT, Multics was in production and she was able to write Fortran programs for a research project.

Titan

This picture is of the operators' area in the Titan machine room at Cambridge University. Notice the hardcopy terminal with paper coming out.

University of Cambridge

Timesharing allows a central computer to be shared by a large number of users sitting at terminals. Each program in turn is given use of the central processor for a fixed period of time. When the time is up, the program is interrupted and the next program resumes execution. This is called time slicing. A program can also be interrupted before the end of the time slice, for example it might have to wait for input. Timesharing is particularly effective when each user wants to use the computer intermittently, such as for program development, which has long periods of editing followed by short test runs. It is not suitable for the batch processing that is typical of commercial data processing or for large-scale computation.

Dartmouth's DTSS and MIT's Multics had different objectives and the systems reflected these differences, but the architectures had many features in common. Multics was closely linked to the emerging discipline of computer science. It is often described as the precursor to Unix. I recall a visit from Brian Kernigham of Bell Labs to Dartmouth in about 1982 when he was intrigued to discover that features of Unix, such as pipes, had equivalents in both architectures. Most of Multics was written in a high-level language, PL/1, unlike other operating systems that were written in assembly code and tightly linked to the hardware architecture.

DTSS had a direct commercial impact. Since large computers were so expensive, commercial timesharing bureaus sold computer time to remote customers. The market leader was General Electric's GEIS, which was originally developed in a joint project at Dartmouth. Concepts from DTSS were adopted by the early minicomputer systems from companies such as Hewlett Packard and Digital Equipment Corporation. Even the computer HAL in the film "2001: A Space Odyssey" had commands taken from DTSS.

Timesharing dominated academic computing until the late 1980s, when it was replaced by personal computers. Even then, the differences between academic and commercial computing remained and led universities to build the first large networks of personal computers. As a result academic

computing and the computing mainstream followed separate paths for about thirty years, from the mid-1960s to the 1990s.

Basic programming

Basic was developed at Dartmouth as a straightforward programming language for students and faculty. It became the language of choice in educational computing. Basic was never suitable for large programs, and for this reason it is often dismissed by computer scientists, but it was ideally suited for its purpose. The simple syntax meant that it was easy for beginners. For example, no distinction was made between integers and floating point; they were just numbers. As editing was always a problem on hardcopy terminals, each Basic statement had a line number; to make a change, the user retyped the line. Finally, the syntax was carefully designed for very fast compilation. Dartmouth specialized in fast compilers but almost every other version of Basic was interpreted.

Because of its efficient use of hardware, the first commercial timesharing systems were based on Basic. Later, for the same reasons, Basic was the dominant language on the early personal computers. Microsoft's first product was a version of Basic. Unfortunately for Basic's reputation, Microsoft Basic had numerous crude extensions to give the programmer control of the hardware.

As computers became more powerful, Dartmouth steadily extended the language. The final version was an elegant structured language, with an effective choice of procedures, and good vector graphics. The design choices always emphasized simplicity. In the early 1980s we used Basic for the introductory computer science course at Dartmouth and PL/1 for the second. For the equivalent courses at Cornell today, we use Python and Java.

Commercial timesharing systems

Some of the first minicomputers ran small timesharing systems with a dedicated Basic interpreter. In 1972 I moved to the British Open University, which was the pioneer in distance education providing degree programs for students across Britain. The university had a national timeshared network using Hewlett Packard's HP 2000 Time-Shared Basic. The university had teletype terminals in study centers around the country. They were connected to the computer system by placing an ordinary telephone handset into an acoustic coupler. The transmission speed was 110 bits per second. The Hewlett Packard computers could each support 32 simultaneous users. There was one computer at the Open University's headquarters in Milton Keynes and I think that there was a second system at a center in the north of England. The computer provided only one service, a Basic editor and interpreter. The version of Basic and the command language were essentially the same as Dartmouth's first version.

The first two computing courses that we introduced at the Open University were based on this system, which was a great success. The main difficulty was not technical. Many of the university's students lived long distances from the study centers and we had to design the courses so that the students did not have to visit the centers very often. My wife was one of the first people to have a terminal at home, in the kitchen. In those days, it was the ultimate status symbol of the working wife.

Digital Equipment Corporation (often known as Digital or DEC) created three very different timesharing systems, each of which was widely used in universities: RSTS for the PDP 11, Tops-20 for the Dec 20, and VAX/VMS. The original RSTS was another Basic system, rather like Hewlett Packard's, but the final version, for the PDP 11/70, was a general purpose timesharing system. It supported up to 63 users and was used by smaller colleges into the 1990s. I never used RSTS, but both the Open University and Carnegie Mellon had Dec 20s. Carnegie Mellon had six of them in the central machine room and the computer science department had several more. Tops-20 had a flexible command language, with a good balance between simplicity and generality. Unfortunately

the unreliability of the hardware often let down the excellence of the software. The Dec 20s had an extended version of Basic but Digital's particular strength was its Fortran 77 language. The VAX built its reputation on the fast computation for science and engineering, but the VMS operating system was also good for timesharing. Many liberal arts colleges used VAX/VMS for their academic computing and in the 1980s it was widely used for departmental computing centers.

Finally, Unix began as a timesharing system at Bell Labs, where it ran on a variety of Digital minicomputers. The version that became a central part of academic computing was the Berkeley Standard Distribution (BSD), from the University of California at Berkeley. The definitive version was BSD 4.2 for Digital VAX computers, released in 1983.

HOME CONTENTS BACK NEXT

May 2014
Please send corrections to wya@cs.cornell.edu