The /360 Architecture and Its Operating System

Frederick P. Brooks
Kenan Professor of Computer Science
University of North Carolina at Chapel Hill
brooks@cs.unc.edu

Ph.D. in Computer Science, Harvard

IBM: Corporate Project Manager

University of North Carolina: Chair of Department of Computer Science

National Medal of Technology, IEEE John von Neumann Medal, ACM Turing Award, Bower Science Award

Member of National Academy of Engineering

Fellow of the American Academy of Arts and Sciences

Major contributions: architecture of the IBM Stretch and Harvest computers and the System / 360 hardware, development of the OS / 360 operating system

Current interests: computer architecture, software engineering, virtual reality

Frederick P. Brooks

The IBM Operating System / 360

The System / 360 Computer Family

From its inception in 1961 until mid-1965, I had the once-in-a-lifetime opportunity to manage the IBM System / 360 Computer Family Project – first the hardware and then the Operating System / 360 software. This computer family, announced on April 7, 1964, and first shipped in February, 1965, defined "third-generation" computers and introduced (semi-)integrated-circuit technology into off-the-shelf computer products.

Context

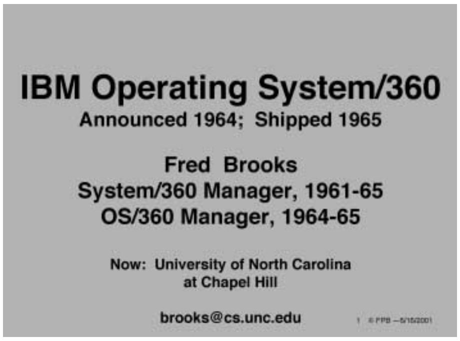
Since many of my audience weren't born then, let me briefly set the context and, in particular, IBM's computer product situation as of 1961. IBM's first-generation (vacuum-tube technology) computers – the 701 large scientific machine, the 702 large commercial machine, and the 650 small universal machine had each evolved a whole family of successors, by 1961 all second-generation (discrete-transistor-technology): 701→7094, 702→7080 III, 650→7074, 650→1620. These had been joined by new second-generation sibling families: the 1401 small commercial and fast

I / O machine and successors, 1401-1410-7010, and the Stretch (7030) supercomputer. Each of these families had reached an end-of-the-line architecturally, chiefly because the instruction formats could not address as much memory as users needed and could afford.

Therefore IBM decided in 1961 to define and build a single new product family intended ultimately to replace all six of the extant families. The family would be the carrier of a new and substantially cheaper circuit technology – wafer-fabricated transistors and diodes. Seven simultaneously engineered models, ranging from the cheapest and slowest to a competitive supercomputer, were undertaken, along with a host of new I / O devices.

A radical conceptual innovation was that all of these models (except the cheapest, Model 20) would be logically identical, upward-*and-downward-*compatible implementations of a single architecture. In software engineering terms, a computer architecture is an abstract data type. It defines the set of valid data, their abstract representations, and the syntax and semantics of the set of operations proper to that type. Each implementation then is an instance of that type. In practice, our first implementations ranged from 8 to 64 bits wide and had various memory and circuit speeds.

Since a computer architecture defines exactly what programs will run and what results they will produce, the product family's strict compatibility enabled us to design a single software support package that would support the whole product family, and could be cost-shared across the entire family, with its large combined market forecast. This in turn enabled building a software support package of unprecedented richness and completeness. In those days, manufacturers gave away operating systems and compilers to stimulate the sale and use of hardware. Within this context, the attached set of slides gives a great amount of detail about this undertaking. Rather than restate all this in prose, I provide elaborating comments keyed to the slides.



IBM Operating System/360

Announced 1964; Shipped 1965

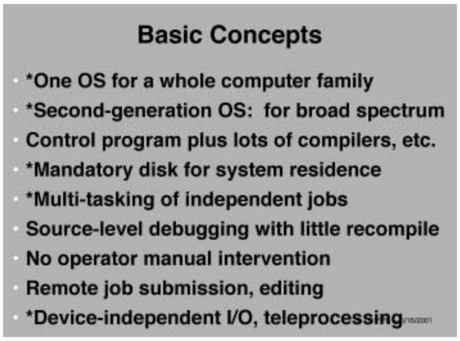
Fred Brooks
System/360 Manager, 1961-65
OS/360 Manager, 1964-65

Now: University of North Carolina at Chapel Hill

brooks@cs.unc.edu

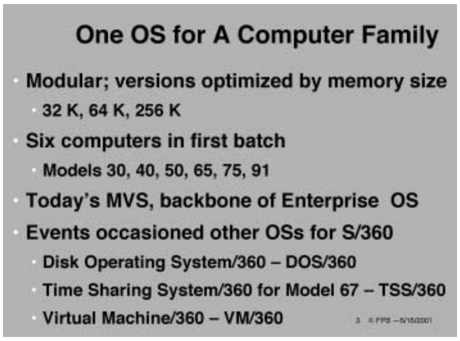
1 © FPB – 5/15/2001

This figure is essentially the table of contents for what follows. The concepts marked with an asterisk are the most important and novel.



Basic Concepts

- *One OS for a whole computer family
- *Second-generation OS: for broad spectrum
- Control program plus lots of compilers, etc.
- *Mandatory disk for system residence
- *Multi-tasking of independent jobs
- Source-level debugging with little recompile
- No operator manual intervention
- Remote job submission, editing
- *Device-independent I/O, teleprocessing

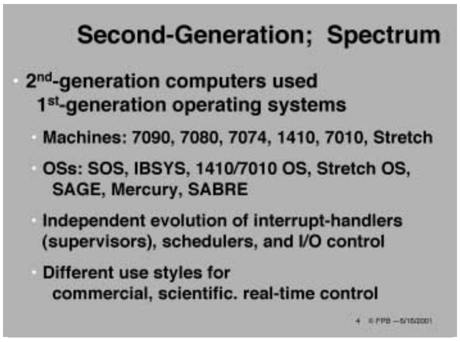


One OS for A Computer Family

- Modular; versions optimized by memory size
  - 32 K, 64 K, 256 K
- Six computers in first batch
  - Models 30, 40, 50, 65, 75, 91
- Today's MVS, backbone of Enterprise OS
- Events occasioned other OSs for S/360
  - Disk Operating System/360 – DOS/360
  - Time Sharing System/360 for Model 67 – TSS/360
  - Virtual Machine/360 – VM/360

3 © FPB – 5/15/2001

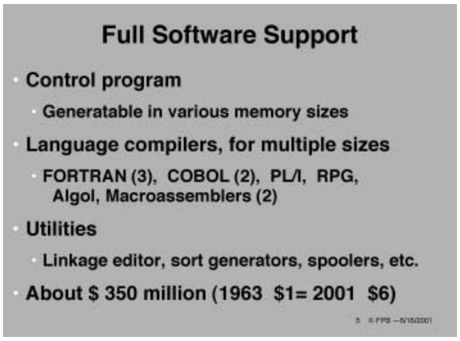
Just as the first-generation operating systems were developed for second-generation computers, so OS / 360 is the first of the second-generation operating systems, and it was developed for the first of the third-generation computers. First-generation operating systems were sharply differentiated by the application areas; OS / 360 was designed to cover the entire spectrum of applications.



Second-Generation; Spectrum

- 2nd-generation computers used 1st-generation operating systems
  - Machines: 7090, 7080, 7074, 1410, 7010, Stretch
  - OSs: SOS, IBSYS, 1410/7010 OS, Stretch OS, SAGE, Mercury, SABRE
  - Independent evolution of interrupt-handlers (supervisors), schedulers, and I/O control
  - Different use styles for commercial, scientific. real-time control

4 © FPB – 5/15/2001



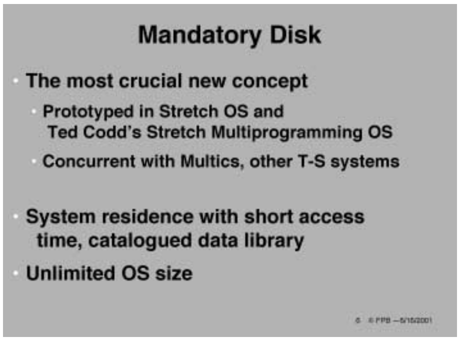
Full Software Support

- Control program
  - Generatable in various memory sizes
- Language compilers, for multiple sizes
  - FORTRAN (3), COBOL (2), PL/I, RPG, Algol, Macroassemblers (2)
- Utilities
  - Linkage editor, sort generators, spoolers, etc.
- About $ 350 million (1963 $1= 2001 $6)

5 © FPB – 5/15/2001

The name *Operating System / 360* is sometimes used to describe the entire software support set. The same name is also used, more properly, to describe the control program alone. We shall use it hereafter only in this more restrictive sense. I like to think of the entire set as one big peach (the control program) and a bowl full of independent cherries (the compilers and utilities). OS / 360 is modular; at system generation one can configure it to contain various functions, and to fit various resident memory sizes and resident disk sizes.

Several Fortran and Cobol compilers, optimized for different memory sizes, were built concurrently. Fortran H (for 256K configurations) was an exceptionally fine optimizing compiler, setting new standards for the quality of global optimization. The team was coherent and super-experienced – I think this was their fourth Fortran compiler.

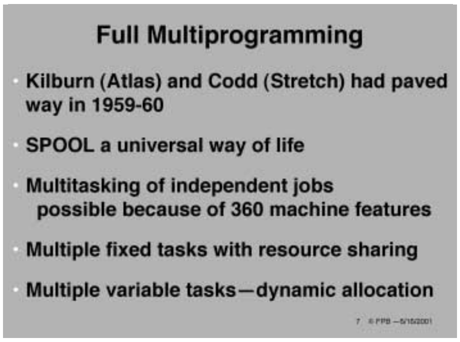


Mandatory Disk

- The most crucial new concept
  - Prototyped in Stretch OS and Ted Codd's Stretch Multiprogramming OS
  - Concurrent with Multics, other T-S systems
- System residence with short access time, catalogued data library
- Unlimited OS size

6 © FPB – 5/15/2001

The new availability of an inexpensive disk drive, the IBM 2311, with its then-immense capacity of 7 MB, meant that we could design the operating system to assume operating system residence on a "random-access" device rather than on magnetic tape, as in most first-generation operating systems. This made the biggest single difference in the design concepts.



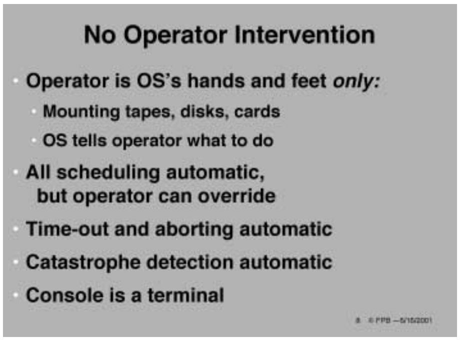
Full Multiprogramming

- Kilburn (Atlas) and Codd (Stretch) had paved way in 1959-60
- SPOOL a universal way of life
- Multitasking of independent jobs possible because of 360 machine features
- Multiple fixed tasks with resource sharing
- Multiple variable tasks—dynamic allocation

7 © FPB – 5/15/2001

Late-first-generation IBM operating systems provided for Simultaneous Peripheral Operation On Line (SPOOL), so that at any given time a second-generation computer could be executing one main application and several card-to-tape, tape-to-card, tape-to-printer utilities. The latter were "trusted programs," carefully written so as not to corrupt or intrude on the main application, which usually ran tape-to-tape. OS / 360 made the big leap to concurrent operation of independent, untrusted programs – a leap made possible by the architectural supervisory capabilities of the System / 360 computers. Early OS / 360 versions supported multiple tasks of fixed sizes for which memory allocation was straightforward. Within two years the MVS version supported multiprogramming in full generality.

A key new concept, now routine, is that the OS, not the operator, *controls* the computer. As late as 1987, some supercomputers, such as the CDC-ETA 10, were still running under operator manual control. A corollary of OS control, pioneered in Stretch and routine today, is that the keyboard or console is just another I/O device, with very few buttons (e.g. Power, Restart) that directly *do* anything.

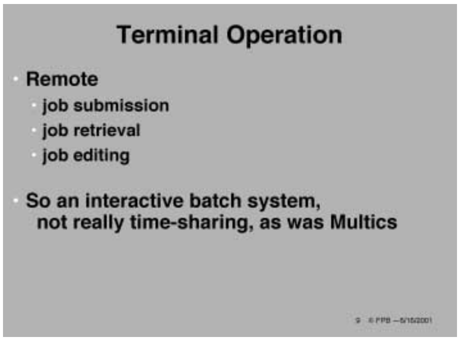


No Operator Intervention

- Operator is OS's hands and feet *only:*
  - Mounting tapes, disks, cards
  - OS tells operator what to do
- All scheduling automatic, but operator can override
- Time-out and aborting automatic
- Catastrophe detection automatic
- Console is a terminal

8 ©FPB —5/16/2001

OS/360 was designed from the ground up as a teleprocessing system, but not really a terminal-based time-sharing system. This concept contrasts with that of the contemporary MIT Multics System. OS/360 was designed for industrial-strength high-volume scientific and data processing applications; Multics was designed as an exploratory system, primarily for program development.

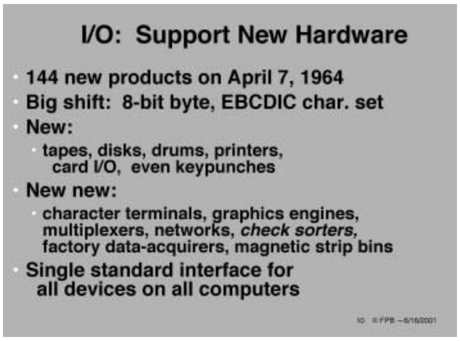


Terminal Operation

- Remote
  - job submission
  - job retrieval
  - job editing
- So an interactive batch system, not really time-sharing, as was Multics

9 ©FPB —5/16/2001

Although strict program compatibility was the most distinctive new *concept* of the System/360 computer family, the rich set of I/O devices was its most important system attribute in terms of market breadth and performance enhancement. The single standard I/O interface radically reduced the engineering cost for new I/O devices, radically simplified system configuration, and radically eased configuration growth and change.

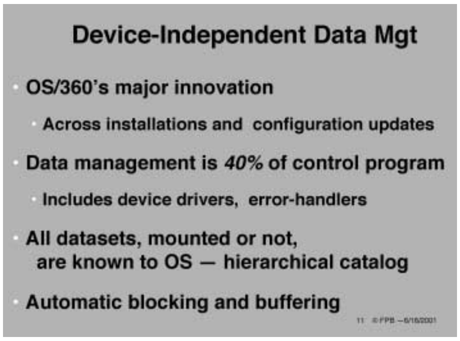


I/O: Support New Hardware

- 144 new products on April 7, 1964
- Big shift: 8-bit byte, EBCDIC char. set
- New:
  - tapes, disks, drums, printers, card I/O, even keypunches
- New new:
  - character terminals, graphics engines, multiplexers, networks, *check sorters,* factory data-acquirers, magnetic strip bins
- Single standard interface for all devices on all computers

10 ©FPB —5/16/2001

The check sorters, curiously enough, posed the most rigid constraint on operating system performance, since they alone of all I/O devices rigidly demanded a quick response – the time of flight between the reading head and the pocket gate is fixed.

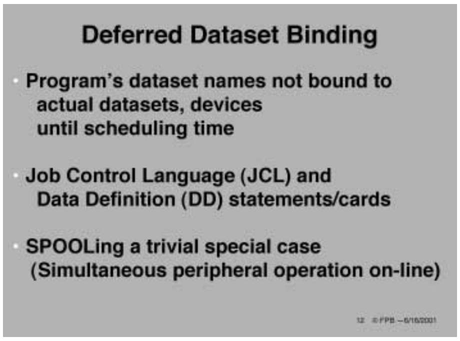


Device-Independent Data Mgt

- OS/360's major innovation
  - Across installations and configuration updates
- Data management is *40%* of control program
  - Includes device drivers, error-handlers
- All datasets, mounted or not, are known to OS — hierarchical catalog
- Automatic blocking and buffering

11 ©FPB – 6/18/2001

The crucial software innovation corresponding to the standard hardware I / O interface was a single system of I / O control and data management for all kinds of I / O devices. This was a radical departure from earlier operating systems. I consider it the most important innovation in OS / 360.

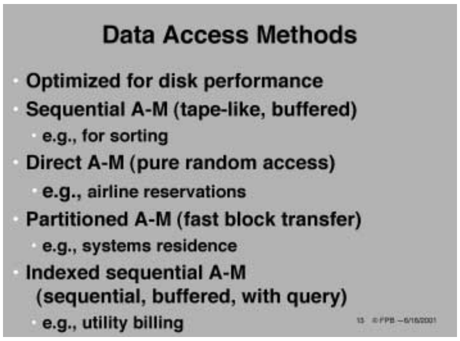


Deferred Dataset Binding

- Program's dataset names not bound to actual datasets, devices until scheduling time
- Job Control Language (JCL) and Data Definition (DD) statements/cards
- SPOOLing a trivial special case (Simultaneous peripheral operation on-line)

12 ©FPB – 6/18/2001

OS / 360, more explicitly than any of its predecessors, recognized scheduling time as a binding occasion distinct from compile time and run time. Not only were program modules bound to each other at schedule time, dataset names were bound to particular datasets on particular devices at scheduling time. This binding was specified by the Job Control Language, which was executed by the scheduler.

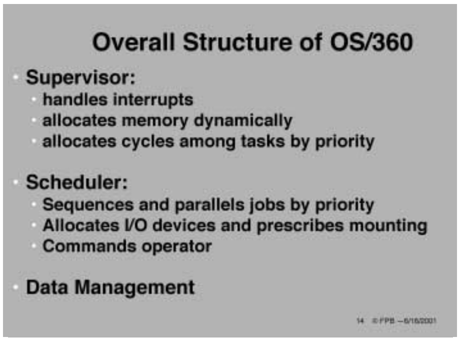


Data Access Methods

- Optimized for disk performance
- Sequential A-M (tape-like, buffered)
  - e.g., for sorting
- Direct A-M (pure random access)
  - e.g., airline reservations
- Partitioned A-M (fast block transfer)
  - e.g., systems residence
- Indexed sequential A-M (sequential, buffered, with query)
  - e.g., utility billing

13 ©FPB – 6/18/2001

Four access methods were designed especially for exploitation of the fleet of new disk types, across the range of disk applications. Two other access methods were designed especially to provide full flexibility and ease of use for telecommunications.

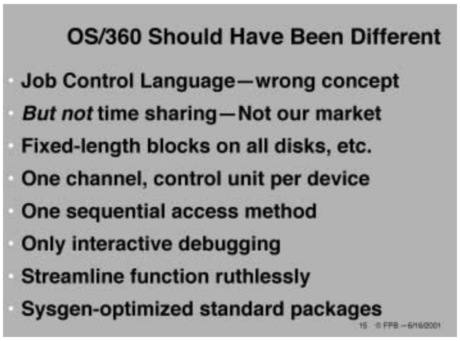


Overall Structure of OS/360

- Supervisor:
  - handles interrupts
  - allocates memory dynamically
  - allocates cycles among tasks by priority
- Scheduler:
  - Sequences and parallels jobs by priority
  - Allocates I/O devices and prescribes mounting
  - Commands operator
- Data Management

14 ©FPB – 6/18/2001

In OS / 360, three independent streams of control program evolution come together. The system structure mirrors this diverse ancestry. The Supervisor evolved from early interrupt-handling routines; the Data Management System from earlier packages of I / O subroutines; the Scheduler from earlier tape-based job schedulers.



OS/360 Should Have Been Different

- Job Control Language—wrong concept
- *But not* time sharing—Not our market
- Fixed-length blocks on all disks, etc.
- One channel, control unit per device
- One sequential access method
- Only interactive debugging
- Streamline function ruthlessly
- Sysgen-optimized standard packages

15 © FPB –6/16/2001

The Job Control Language is the worst programming language ever designed by anybody anywhere – it was designed under my management. The very concept is wrong; we did not see it as a programming language but as "a few control cards to precede the job." I have elaborated on its flaws in my Turing lecture [2].

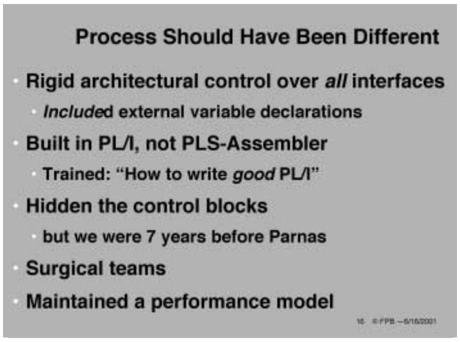
We should have cut loose from the key-count-data variable-length block structure established for IBM's earlier disks and designed for one or two sizes of fixed-length blocks on all random-access devices.

The I / O device-control unit-channel attachment tree is unnecessarily complex [3]. We should have specified one (perhaps virtual) channel and one (perhaps virtual) control unit per device. I believe we could have invented one sequential disk access method that would have combined the optimizations of the three: SAM, PAM, ISAM.

Two radically different debugging systems are provided, one conceived for interactive use from terminals with rapid recompilation, the other conceived for batch operation. It was the best batch debugging system ever designed, yet totally obsolete from birth.

OS / 360 is too rich. Systems residence on a disk removed the size constraint that had disciplined earlier OS designers – we put in many functional goodies of marginal usefulness [4, Chap. 5]. Featuritis is even now not yet dead in the software community.

The system-generation process for OS / 360 is wondrously flexible and wondrously onerous. We should have configured a small number of standard packages to meet the needs of most users, and offered these to supplement the fully flexible configuration process.



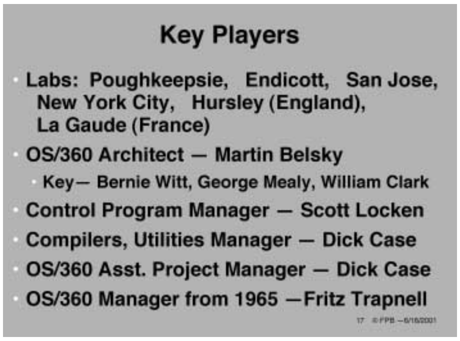
Process Should Have Been Different

- Rigid architectural control over *all* interfaces
  - *Include*d external variable declarations
- Built in PL/I, not PLS-Assembler
  - Trained: "How to write *good* PL/I"
- Hidden the control blocks
  - but we were 7 years before Parnas
- Surgical teams
- Maintained a performance model

16 © FPB –6/16/2001

I have treated this topic at length in *The Mythical Man-Month* [4]. Here I will only elaborate a little. I am firmly convinced that if we had built the whole thing in PL / I, the best high-level language available at the time, the OS would have been just as fast, far cleaner and more reliable, and built more swiftly. In fact, it was built in PLS, a syntactically sugared assembler language. Using PL / I would have required careful training of our force in how to write *good* PL / I code, code that would compile to fast run-time code.

We should have maintained rigid architectural control over all the interfaces, insisting that all declarations of external variables be *included* from libraries, not crafted anew in each instance.



Key Players

- Labs: Poughkeepsie, Endicott, San Jose, New York City, Hursley (England), La Gaude (France)
- OS/360 Architect — Martin Belsky
  - Key— Bernie Witt, George Mealy, William Clark
- Control Program Manager — Scott Locken
- Compilers, Utilities Manager — Dick Case
- OS/360 Asst. Project Manager — Dick Case
- OS/360 Manager from 1965 —Fritz Trapnell

17 ©FPB –6/18/2001

About 1000 people worked on the entire OS / 360 software package. Here I identify both the teams and those individuals who contributed most to the conceptual structure.

References

[1] IBM Systems Journal, 5, 1 (1966)

[2] Brooks, F.P. Jr., “The Design of Design.” ACM 1999 Turing Award Lecture, Communications of the ACM, to appear.

[3] Blaauw, G.A., and F.P. Brooks, Jr., 1997. Computer Architecture: Concepts and Evolution. Addison-Wesley, Boston, Section 8.22

[4] Brooks, F.P. Jr., 1995. The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition. Addison-Wesley, Boston.