Operating System Roots

By Robert L. Patrick

Written: December, 2006

CHM Reference number: R0369.2017

Author Sketch: U of Nevada, engineering

USAF

1951-1959: Convair, GM Research, CEIR, CSC

1959-1992: Independent consultant

1992: Retired

Abstract: Robert L. Patrick describes the various computer systems he worked on in the early years of computing. His review covers the IBM Card Programmed Calculator through the IBM 701 with the Speedcode interpretive programming system through the IBM 704 with an early FORTRAN compiler. This was followed by the IBM 709 with the SOS operating system, developed jointly SHARE and IBM, and then the IBM 7090 with the IBSYS operating system developed by IBM. Successful features from all of these operating systems were incorporated into the design of System/360 when it was released in the mid-1960s.

Introduction

To understand operating system software, it helps to go back to the beginning. In the late 1940s, some engineers at Northrop Aviation (now Northrop-Grumman Corp.) cable-connected an IBM tabulator to an IBM calculating punch and the Card Programmed Calculator was born. IBM delivered CPCs in kit form consisting of manuals, big boxes of hardware, blank plugboards, and about 50 pounds of wires of various lengths. In 1949, IBM sponsored a seminar to discuss using the CPC as a general purpose, externally programmed, engineering computer. About this same time a team under Ev Yowell (at the National Bureau of Standards center on the UCLA campus) devised a set of three-address, decimal plugboards to support engineering calculations on the CPC.

I met this combination in 1951 when Vern Kamm (another USAF 2nd Lt.) and I worked a week at UCLA to wire a set of plugboards for the soon-to-be-installed CPC at Edwards AFB in the California desert. In addition to the (tested) plugboards themselves, the team at UCLA provided us with a deck of test cards which used the boards and exercised all the necessary circuits in the CPC. When the machine was installed and this test deck successfully ran, we were in business.

Edwards was the principal center of USAF airplane flight testing and the observed data from each test flight had to be pushed through an appropriate set of formulas to be useful. Prior to the CPC, this data reduction was done by engineers with long slide rules. After programming the CPC, computed results were available overnight from the small computer center run by enlisted personnel. I did the programs, set up the center, and trained the operators.

In 1953, my USAF service was over and I moved to Convair in Ft. Worth, TX. Convair had a backlog of engineering computing, and their work was run on CPCs operated remotely by the accounting department. I programmed and in some cases wired plugboards and sometimes operated these CPCs until our engineering IBM 701, serial number 7, arrived.

Meanwhile, John Backus and a crew at IBM prepared an interpretive system for programming and operating a 701. It was called Speedcode. It programmed just like the Yowell CPCs except it had floating point arithmetic. (What a blessing to have no more scaling.) I moved easily onto Speedcode and started providing engineering computer services while most of my colleagues struggled with subroutines, scaled binary, assembly languages, and debugging in binary/octal/decimal.

Soon the 701 was full of work. The style of operation was 'programmer-present and operating'. Each programmer was an entity unto himself to choose his software tools, get ready, sign up for test time, and run the tests. The 701 would only support about 18 programmers this way.

Speedcode supported magnetic tapes and I found I could load from tape, pick up some program corrections/additions/changes from the card reader, run a test shot, and print out my results while the next programmer on the schedule was getting set up to make their run. Along the way I read an article on time and motion studies (later published in *Gantt on Management*, A.W. Rathe, AMA, 1961) and applied his principles to computer operations.

In 1954, I moved to General Motors Research which had CPCs and a 701 about to be installed. Shortly thereafter we started planning for the (then new) IBM 704, which was faster and had floating point arithmetic built into the hardware. But since the architecture of the 704 still used the CPU for all functions, it could do only one thing at a time (e.g. read, compute, or write). However, IBM also offered three standalone peripheral machines which would read cards and write magnetic tape, read tape and print, and read tape and punch cards; all completely independent of the mainframe.

These fit the style of operation I had been experimenting with at Convair and the three-phase operating system for the 704 resulted: The peripheral operator would take all jobs that had been submitted and load them on magnetic tape. The tape would be physically handed to the mainframe operator in the next room. When the 704 was available, the tape would be hung and the operating system would take over. Each program was surrounded by a series of control cards which would define the processing to be done. As these card images were interpreted by

the system, the proper software would be called from the system residence (Sysres) device (a drum) and the software and the programs would be mated. When that step was completed another control card image would be read and the process would be repeated until the input tape was exhausted. As each program progressed, it would write output on a magnetic tape in binary for later translation to decimal and printing offline on the peripheral printer.

The Sysres device held a decimal-to-binary converter, the SAP assembler (by Roy Nutt of United Aircraft), and an early FORTRAN compiler (by John Backus and another IBM team). It was possible to 'load and go' and the mainframe operator had a standard end-of-job dump routine to follow in case of an abnormal end.

Production and test were routinely mixed in the input stream since the machine retained no knowledge of the previous job once the END card had been processed.

I wrote up a description of this proposed operating system and presented it at SHARE 3 in Boston. Owen Mock of North American Aviation (now Rockwell) had been thinking along the same lines and our managers decided to do the system jointly. Owen and his crew did the input end and the output translators were produced by a GMR team led by George Ryckman (later a SHARE president). George also devised hardware for a time-of-day clock which we used for operator run-time information and job accounting. When the system was in full operation, GMR offered desk-to-desk courier service so that programmers could concentrate on preparing new work. With the old signup sheets and programmer present operation, ten average jobs per hour was a full load. With the three-phase batch system we got ten times the throughput with no increase in rental.

The 704 was only offered in a small number of memory sizes and I/O configurations. The GM-NAA system supported only these configurations (through fixed addressing) and made no attempt at generality.

In 1957 IBM announced the 709. Its architecture could be described as a 704 with channels. In IBM parlance, a channel is a limited purpose computer that shares access to the memory bus with the CPU, and sits astride the information that flows between the memory and the set of input/output devices the customer has installed. When configured with a channel, the CPU and the channel are synchronized only for a few microseconds while the CPU instructs the channel what to do. Then the CPU is free to process data while the channel selects the device and transfers other data to/from memory.

Development of SHARE Operating System

A team of programmers from SHARE installations and IBM built a new operating system for the 709. It was called SOS (SHARE Operating System). It exploited the channel architecture of the 709, benefited from the work done by GM and North American, and added features to support a wider variety of workloads.

When the 7090 came out it was sufficiently different that the next operating system in the series was required. IBSYS was mostly done by IBM. In addition to the expected improvements and extensions, IBM added features to make it more appealing to their intended customers. IBSYS supported a large number of different configurations, so the salesman was free to offer just what the customer needed. It was a good system and contained extensive features to provide a wider variety of memory sizes and I/O devices, including support for a large fixed disk file.

One of the limitations of these early systems was the original batch concept. With these systems, jobs were gathered into batches and the first job in was always the first job out. With big batches and long running times, business priorities could change after the job was submitted. Programmers with top priorities could not afford to wait for the entire batch to complete before getting their output. While customers struggled with this administrative problem, IBM announced the 7040. A 7040 had the architecture of a 7090 and could run IBSYS. Further, it was priced cheaper than a pair of 7090 channels.

When IBM announced a minor product that would allow two computers to communicate channel-to-channel, the scene was set for a major extension to IBSYS. This was called the Direct Couple, and consisted of a pair of machines with both running a modified version of IBSYS. The senior computer could be a 7090, 7094, or 7094 II, and the junior machine could be a 7040 or 7044 (and in rare cases another 709X).

The Direct Couple allowed existing customer jobs to run on the senior machine while the junior machine handled all the I/O. Jobs were introduced to the system via the junior machine, were queued on the disk, were selected for execution from a queue based on current priority, then executed on the senior machine, and finally each job sent its output back to a queue on the disk. At end of job on the senior machine, another priority queue was referenced and the most desired output was printed first.

A large variety of configurations was supported by this system including various mainframe speeds, memory sizes, and a menu of I/O including remote Input/Output Stations connected via communications lines (on-site or across the country). The Direct Couple extension to IBSYS was put together by a programming team from Aerospace, NASA-Houston, and IBM.

Release of OS/360

Finally came OS/360. It was designed in 1962, announced in 1964, and delivered in 1966. It was done entirely in-house by IBM (with maybe some contract help). It continued and extended all the features discussed above plus some very nice architectural extensions which aided both IBM and their customers. Although the System/360 hardware architecture was a complete break with the past, most of the proven features from these early operating systems were carried over.

The System/360 had bounds registers in the hardware to fence the operating system software off from the running application programs. This guaranteed the OS would survive even if an application program failed. Further, OS would allow several application programs to share memory and be allowed to run while some other program was awaiting the completion of I/O. The SYSGEN process would allow (in theory) revised system software to support hardware configuration changes without changing any programs in the application library. While OS/360 had development problems and the early versions of the Attached Support Processor (ASP) were slower than a finely tuned Direct Couple (IBSYS) System, it was a giant step forward in the art of operating systems.

In the late 1960s, a team, once again made up of programmers from customer installations and IBM, added an important extension to OS/360. This extension was called IMS/360 (Information Management System/360). It was a System/360 improvement of a set of software Rockwell had made for the IBM 7010. It provided data management services and controlled multiple terminals connected to a small number of applications as batch application programs ran in the background.

The operating systems that provided multiprocessing were huge and complicated, but they simultaneously supported remote terminal services, applications development, and applications production on a single set of hardware. Further, application programs could fail without affecting other applications programs in the load, multiple computers could share a single load of related work, and the system could recover from program failures and rebuild the database (usually) with minimal human intervention.

In the world of big iron, the GMR-NAA operating system was a precursor to SOS, IBSYS, the Direct Couple, and the collection of services known as OS/360.

