My High-Tech Adventure

Computing As I Knew It 1960-2024 (Third Edition)

Contents

Legal, Copyright

My High-Tech Adventure: Computing As I Knew It 1960-2024 (Third Edition) is Richard H. (Dick) Johnson's memoir about his experiences working in the computer industry. The first edition covered the years 1960-2003. The second edition includes an epilogue that covers the years from 2003 to 2012.

Chapter 1. Introduction

In this brief history I have recorded some personal memories of my experiences working in the computer industry. This is not a history of computing (there are plenty of those already!), but rather a description of things I actually had experience with myself. The period covered is from in the 1950s through 2012. I have written this because I think a “remembrance” of computing might be more interesting to some people than just a history. Computing has changed so much in the years I have been in it, that it might be hard to for some to believe the world of computers did not always revolve around Microsoft Windows and the Internet.

The primary audience for this book is my friends and family. I hope they find it interesting!

My years in elementary school through my graduation from college coincided with the initial development of commercial computers and the birth of the computer industry.

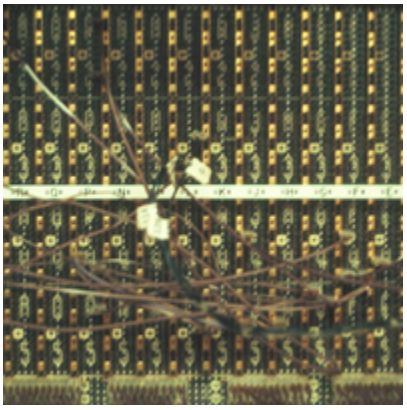
The following table lists the dates I was in school and some of the most popular IBM computers at the time. This information will help to set the context for the next few chapters.

Note:
During these years there were also computers marketed by other companies like CDC (Control Data Corporation) and UNIVAC, but my experience was mostly with computers from IBM.

Year	Computer type, model	Description
Elementary school and junior high (1948-1956)		
1952	IBM Defense Calculator	Early binary vacuum tube computer
1952	IBM 701	Early manufacturable computer
1955	IBM 704	First successful core memory mainframe
1955	IBM 650	Computer with magnetic drum memory
High school (1956-1960)		
1957	IBM 709	Last vacuum tube mainframe
1958	IBM 7090	Early transistor computer
1959	IBM 1620	Early small scientific computer
University (1960-1965)		

Year	Computer type, model	Description
1959	IBM 1401	Early business computer
1962	IBM 7094	Large scientific transistor computer
1965	System/360	First models of new computer family delivered

The figure below shows an early computer backplane.



Chapter 2. Early experiences with computers

My earliest experience with "computers" was a tic-tac-toe machine I built when I was about 10 years old. It never really worked.

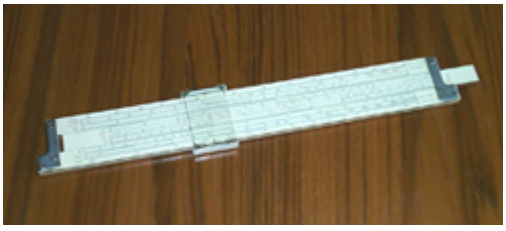
My tic-tac-toe machine

When I was about 10 (1952), I tried to build my own computer that would play tic-tac-toe. This “computer” was made out of plywood, switches, wires, light bulbs and dry cells, and it never really worked. I spent the money my parents gave me for a birthday present buying the parts at the hardware store. I put in many hours working on it in the basement of our house. That was probably the first time I had anything to do with computers. I also learned how to use the abacus and slide rule. My friends, Philip Mitchell and Lane Carroll, and I used to play games calculating numbers like 9 to the 9th to the 9th power just to amaze ourselves.

Early interest in personal computers

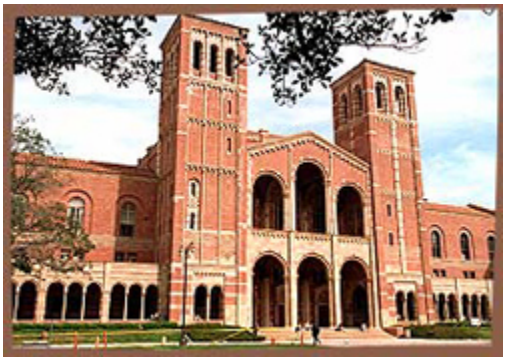
When I was in junior high school, I first began to hear about digital computers. If you consult a history of computing, you will find that the time I was in junior high (1955-1956) was just when computers began to be used in business and the government in a big way. One day in 1959 my mother heard my friends and me talking about something related to computers, and she actually phoned the local IBM branch office to ask if they had any inexpensive computers that she could buy for me. At the time the cheapest thing probably cost several million dollars! So my mother had the idea of the personal computer long before the industry did.

The following figure shows the "computer" I used in school.



Chapter 3. Computers at UCLA

"Machines should work. People should think." —IBM Pollyanna Principle



Western Data Processing Center

I entered UCLA as a freshman math major in the fall of 1960. At that time, IBM and UCLA were joint participants in running a data center at UCLA. The center was located in the Graduate Business Administration building on campus. Shortly after arriving at UCLA, I went on a tour of the Center. This was the first time I had seen a real computer "up close and personal." It was an IBM 7090 with assorted card readers, printers, and tape drives attached to it. The tour guide keypunched a short Fortran program that calculated a table of sines and cosines for us and printed them out on the line printer attached to the machine. The speed with which the machine did this and the flashing lights on the 7090 console fascinated me. When we entered the glass-walled machine room, the card reader was on the left, the 7090 in the middle, and the line printer on the right. The guide put the cards in the reader, which went "kachunk, kachunk!" reading the cards, then the lights flashed on the 7090 console for a few seconds, and almost at the same moment the line printer began printing the results. Prior to this I had been used to calculating sines and cosines with tables of logarithms and slide rules, which took a lot longer.

The person who led the tour was Steve Crocker, who later went on to become famous designing the ARPANET. The Center housed the machine in a setting that was fairly typical of early machine rooms. The front wall of the room was all glass, which allowed people passing by to see the flashing lights and moving tape drives. This is where the term "glass house" came from.

The following figure shows an IBM 7090 computer.

UCLA Computer Club

In 1960, the way a student learned about computers at UCLA was to join the UCLA Computer Club and take Saturday classes from them. There were no computer science classes or majors at the time. I majored in mathematics, which was the closest thing I could find to computing. Also, I really was very interested in mathematics. My first computer class was “Introduction to Fortran Programming on the IBM 7090.” At that time, this meant learning the Fortran II language, the first of many computer languages I learned. The first Fortran program I wrote calculated and printed a table of compound interest. The programs were key-punched on an IBM 026 or 029 key punch onto 80 column cards. The card decks could be listed on an IBM 407, which read in card decks and printed them on a printer. We would place our card decks in a long metal tray kept in the basement of the Mathematics Building, and our programs would be run overnight on the IBM 7090. If we were lucky, we would get the printed or punched output back the next day. Sometimes we got nothing at all, because the computer had been busy doing something “more important.” This meant you could wait a day or two just to find out you had made a key punch error!

My first computer program

The program listed below produces a table of compound interest values. The first two lines are control cards identifying who gets charged for the run and that the Fortran compiler is to be invoked.

```
*9017J 2 10 RICHARD JOHNSON UCLA COMPUTER CLUB HOMEWORK PROBLEM
*FORTRAN
      DIMENSION PV(13,100)
      DO 10 N=1,100
      DO 10 I=1,13
      XI=35+5*I
      PV(I,N)=1./(1.+XI/1000.)**N
10    CONTINUE
```

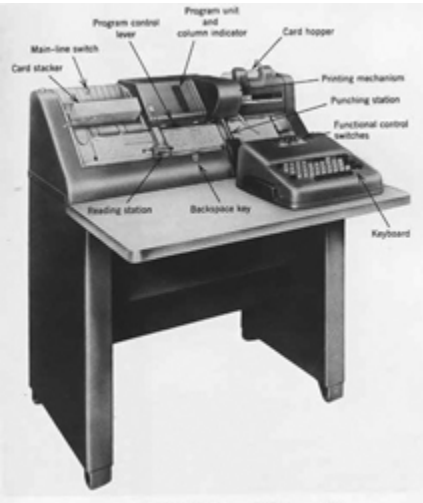
```
      WRITE(6,11)((PV(I,N),I=1,13),N=1,100)
      STOP
11    FORMAT(1X,13F9.6)
      END
```

The 7090 took about 7 seconds to run this program, and my latest Thinkpad laptop computer runs the same program in .01 seconds.

Not long after I wrote this program, I wrote another one like it and submitted it for an overnight run. The next day I got the results back, and they were wrong! I was convinced that the 7090 had made a mistake. It was only a day or so later that I found I had created my first software bug.

Since then I have created thousands more.

The following figure shows an IBM 026 key punch.



Computer operator

One time in 1960 I came home to Lemon Grove from UCLA on a visit, and my friend Lane Carroll took me to the place where he was working as a computer operator. I don't remember what the place was (maybe one of the aerospace companies?), but Lane was the operator for an IBM 1620. This was the small scientific machine of its day (even though it didn't have a divide instruction!) It had 20K of memory and was pretty slow. But it was relatively cheap, so people bought it. Lane had programmed it to type out random doggerel poetry stanzas, and we had great fun watching. That was the first time I actually got to operate a computer. In those days it was not easy to get near a computer, since they were all expensive and scarce.

The figure below shows an IBM 1620 computer.

Smyth Research Associates

At UCLA I interviewed with Dr. John Smyth for a summer job after my freshman year. I worked that summer and the next at Smyth Research Associates in San Diego. The company's only customer was the U.S. Air Force. Smyth specialized in the physics of the bending of radio waves by the ionosphere, and the Air Force was interested in this because they wanted to track radio signals from Soviet missiles and wanted to know how to account for bending effects in the atmosphere. I spent both summers doing bending calculations on a Friden desk calculator, because at that time it was cheaper to hire a student to do this than to buy time on a computer to do the same calculations. In my junior year at UCLA I wrote a small Fortran program to do these calculations and it took seconds to do what had taken me and the Friden hours.

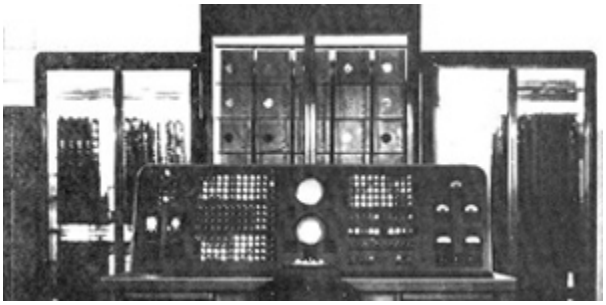
The following figure shows the front of Smyth Research Associates in San Diego, California.

Graduate school

After I got my BA in mathematics in 1964, I enrolled as a mathematics graduate student at UCLA. My advisor was Dr. Charles B. Tompkins. Tompkins played a significant role in the invention of linear programming after World War II. Dr. Tompkins wanted me do one research project using the SWAC (Standards Western Automatic Computer), which was housed in the basement of the Mathematics Building. SWAC dated from about 1952, and it was a vacuum tube machine that was not very reliable

by the time I saw it in 1965. I never actually got to write a program for the SWAC, but I did get to see it being used by other students. Just before I got my BA from UCLA, I decided to try to work in the computer industry. Dr. Tompkins got me an interview at IBM on Wilshire Blvd. I talked with someone there who told me to take numerical analysis classes if I wanted to work at IBM after college. I got my MA in mathematics in 1965, and I did take numerical analysis.

The following figure shows the SWAC.



Chapter 4. IBM Service Bureau

I enter the computer business.

My first job in computing

My first job after I got my MA in 1965 was to be a “scientific application programmer” at SBC (Service Bureau Corporation) in Los Angeles at their Scientific Center. SBC also had other scientific (and commercial) offices in other cities. SBC was a wholly owned subsidiary of IBM that had been created to allow IBM to satisfy the terms of an antitrust consent decree between it and U.S. Department of Justice. At that time, Control Data Corporation (CDC) was a major competitor for IBM, and CDC had claimed that IBM was “bundling” computers and programming (software) to give it an unfair advantage, so by the terms of the consent decree, IBM had to split the programming of computers off into its own company.

When I first got there, SBC sent me to a six-week class on the IBM 1401 and the IBM 7094 computers. These were the two machines they had in their machine room. My instructor was Harry King, who knew both machines as well as anybody could, and was a fantastic teacher. In the class, Harry taught us the complete instruction set for both machines, plus the Fortran language and the FMS and IBSYS operating system environments. We did daily programming exercises that were key-punched by the students and run on computers at SBC. These classes were very intense, and I enjoyed them a lot. I now had no doubt that computing was where I wanted to work. I was the top student in the class.

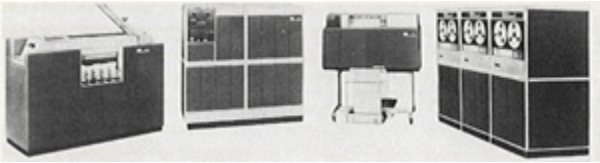
Before going further, I’ll describe the hardware and software of the day.

IBM 1401 hardware and software

This machine was about the size of a large freezer. IBM sold thousands of them and made a lot of money from it. The SBC 1401 had between 4K and 16K of memory. It was a variable word-length machine, which made programming it tricky. The words were delineated by things called “word marks.”

At SBC, the 1401 was mainly used for reading card decks onto tape and for printing tape output from batch jobs run on the 7094. You could do simple commercial applications on the 1401, but SBC didn’t use theirs for that. SBC ran a little program called the “4K Monitor” on the 1401 that provided the functions of card-to-tape, tape-to-print, card duplication, etc. The most common language for writing programs for the 1401 was called Autocoder, which I learned. If you tried to do anything very compute-intensive, the machine would slow to a crawl. By modern standards, it was a very slow machine. I remember having a bug in one of my 1401 Autocoder programs that overlaid a work mark and then printed an entire box of blank paper before the operator noticed what was happening. That was really embarrassing!

The following figure shows the IBM 1401 computer.



The third box from the left in the 1401 configuration above is an IBM 1403 line printer, which was the standard printer we used for output from runs on the 7094. These printers were impact printers that printed on form-fed paper by striking a revolving metal chain with a small hammer at just the right time to print a particular letter.

The revolving print chain and the hammer strikes made a very loud high-pitched noise that was only partially muffled by the cover. You could raise the cover under control of software to allow for changing printer forms. People used to get in big trouble by setting a cup of coffee on top of the 1403 and then having it dumped over when the printer cover was raised!

IBM 7094 hardware and software

The 7094 was SBC's mainframe. It was one of the bigger, faster scientific machines at the time. It could add two numbers in a few microseconds. Memory was 32K of 36-bit words (about 128K today). It stored characters in 6 bits in a code called BCD, so each word held 6 characters. This led to things like variable names in Fortran being limited to 6 characters (one word) in length.

The 7094 did not have user/supervisor state. This meant that programs could, and did, clobber the operating system. Because of this, it was the custom to reload the operating system from tape after each batch job completed.

The 7094 also did not have memory parity checking. When main core memory was not working right, it could make loading and storing mistakes without telling us about it. We could usually tell when memory was failing, because the Fortran compiler would begin producing strange error messages. We used to watch for those. We knew certain messages meant we had bad memory cores.

Almost all programs run on the 7094 at my SBC office were written in Fortran II. Fortran was one of the first algebraic higher-level computer programming languages. Fortran was also one of the first language compilers.

The Fortran language originated in the middle 1950s. Fortran program statements looked a little like algebra, for example: `X=Y*SQRT(Z).`

SBC also had a commercial division, where all the programs were written in COBOL. I never learned COBOL.

SBC normally ran two operating systems on their 7094: FMS (Fortran Monitor System) and IBSYS. FMS was primitive and had just enough capability to load, compile and run Fortran II and 7094 Assembler programs. All input data was on cards or tape, usually tape. The machine had about a dozen 7-track tape drives attached to it. IBSYS was a little more sophisticated than FMS, because it had an I/O Control System built into it. This was like a simple-minded file system.

The 7094 had no disks, no terminals, no graphics displays, and no network, but lots of flashing lights.

The following figure shows the IBM 7094 computer.

Note:
The suits you see people wearing in the above picture are typical of the time. These machines were very expensive and hence were treated with respect!

Working environment

SBC had a 1401 and 7094 II in one large room with a glass wall across the front. SBC's main business was writing and running scientific computer applications and renting 7094 computer time. SBC made most of its revenue renting machine time. The 7094 rented for $600/hour. Machine time was measured by stamping a timecard at the beginning and end of each job on a time clock. I was part of a 15-person programming staff. The operations staff was about the same size. Our branch office was about a mile from Los Angeles International Airport. We could watch the airplanes taking off and landing out our office windows.

Our staff (including me) used to wear suits and ties at all times. Our main 7094 operator was a particularly fancy dresser and wore a large diamond pinky ring. Nobody ever came to work wearing casual clothes, even late at night or on weekends.

When I first got to SBC, my office was in a building a few blocks away from the machines. When we wrote programs, they got keypunched by our keypunch operators and then the decks were taken by courier to the main building to be run on the machines. If we were lucky, we might get 2-3 “turnarounds” per day. We would have to wait hours just to find out we had made a typo! We got really good at visually checking our card decks for errors the machine might hit. When I was new at SBC, I had to write program flow charts before I even key-punched the program. The flow charts were checked by others. After I had been there a while, they quit asking me to do flow charts.

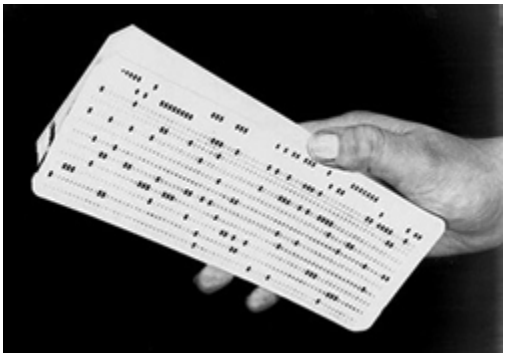
Programs were all stored as 80-column card decks. Once a program was debugged, it was kept for us in a file drawer by Jean, our program librarian, who stored all the programs in long file drawers. There was a catalog of the programs kept in a three-ring binder.

Card decks sometimes got dropped. In order to facilitate getting the cards back in order, we either sequenced the deck in the right-hand columns of the cards or we drew a diagonal line across the top of the deck with a marker pen.

“Jobs” were card decks with operating system control cards at the front followed by the Fortran source code and finally any input parameters. The cards were read on the 1401 and copied card-to-tape to produce a “job tape.”

When it was time to run your job, it was taken to the 7094 and the OS was loaded from a tape drive with your job on another tape drive. Output from your job was written to yet another tape drive. After the job finished, the output tape was taken back to the 1401, which used a tape-to-print utility to produce the printed output. You got back either output from your program or a core dump, which was memory printed in octal. The core dump meant your program had a bug and had crashed the operating system on the 7094. We would stand by the 7094 printer console when we had an important job to run and could tell quickly whether the job was running normally or had ended in a dump. As a courtesy, the operator would phone us in our office so we could attend critical job runs in the machine room.

The following figure shows an 80-column IBM card.



My first failed project

Soon after I got to SBC, I was assigned to help a senior programmer debug and run a Fortran program that did grade reporting for several local Catholic high schools. The senior programmer got the code to the point where it compiled and ran, and then he disappeared for several months to attend the IBM SRI (Systems Research Institute) in New York City. After he had left we got the first live data from a local school and found out on the first run that things were not working right. I was the only programmer available to work on fixing the code, and I was a brand-new employee. After a few days I found out that almost every single subroutine in the code was full of bugs, and I had to rewrite and debug almost everything. Many long nights later we finally were able to give the customers results, but it was much too late to be useful. By the next semester we could produce results on time, but the cost to SBC was much more than we were able to charge the customers. In those days using 7094 for such a task was truly like using a sledge hammer to drive a tack. The contracts were not renewed after that first year.

For me the good thing was that I was able to show SBC that I had talent for writing and debugging software. I got promoted several times because of this “failed” project. From this I first learned that a “bad” situation might not always be bad for me personally!

Dr. Chuck Fillerup

After this failed project I worked under the direction of a mentor, Dr. Charles (Chuck) Fillerup. Chuck had been in the computer business for several years before I met him. He was one of the early pioneers. His specialty was numerical analysis and linear programming. Chuck used to talk a lot about how much “fun” he had programming the IBM 650. This machine had been introduced by IBM in 1955. It stored data and programs on a revolving magnetic drum. Programming it was very time critical. Under Dr. Fillerup I worked on several projects, including:

- CRISP: a program that calculated the critical speed of a rotating shaft. A critical speed is one where the shaft will tend to vibrate.
- A program that calculated loading on an air slider bearing (like read/write heads found in disk drives). This project was done for National Cash Register (NCR), then a significant computer company.
- Seat equivalence optimization for the Los Angeles Civic Light Opera using mixed integer linear programming. The Light Opera was opening a new theater and wanted season ticket holders in the new theater to have seats as good as what they had in the old theater.

This is Chuck in the following figure.

I got a few letters from Chuck as late as 1972. He tried to convince me to go to work for Computer Sciences Corporation (CSC) in LA, where he had moved himself. His letters were usually from Milan, Italy, where he was working on a contract. I found out just recently that Chuck died in September 2002 (age 73) after having spent his last 7 years as a part-time member of the faculty at Cal State Long Beach. He had also been teaching Navy personnel in the Persian Gulf aboard ships. His students all said he was quite a guy. One thing he told me that I still remember; “If you stay some place long enough, they always take you for granted.”

Chuck did not like being taken for granted.

Managers

My first manager at SBC was Tony Landi. He suffered through the failed grade reporting project with me. Years later I met Tony again in Kingston, NY, where he was a manager in IBM.

Our SBC office manager was Bob Maldonado. Bob taught me a good lesson I still remember: I asked him for permission to work on a small skunk-works project while I was between customer jobs. Bob told me, “Don’t ask for permission unless you need to! You might get told no, and then what do you do?."

My last manager at SBC was Ken Tranbarger, who came to SBC from Litton. Ken taught me to write GPSS simulations in only a few days after SBC got a simulation contract from Litton. I wrote a simulation model of something similar to the AWACS airplanes of modern warfare. One strange thing about this project was that Litton had us changing constants in the model until the simulation got the right answer. I thought the model was supposed to show whether the proposed system would work. Litton just wanted to get the contract!

The coming of System/360

After I had been at SBC for about a year, they took delivery of a small IBM 360 model 30 computer. The IBM System/360 was an entire family of machines that had the same instruction set. The model 30 was one of the smaller models that SBC wanted to use to replace their 1401s. This machine had 64K of memory and a few tape drives attached to it. The model 30 ran a primitive operating system called BPS (Basic Programming Support). We used BPS, because IBM did not yet have OS/360 running. There was a period of time after it was first delivered when I got to operate the model 30 all by myself. This was my first “personal computer.”

In 1966, SBC replaced their 7094 with an IBM 360 model 40 coupled to an IBM 360 model 65. The model 40 did all the front-end work of reading in cards and printing output from the 65. The 65 did all the real computing. The software consisted of an early release of OS/360 and a software system called ASP (Attached Support Processor).

Attached to the model 65 were several 2311 disk drives (holding 7.25 megabytes each!). These were the first disk drives I had ever seen and I remember wondering how to use them instead of tape. The 65 had firmware that allowed it to emulate (theoretically) a 7094. We soon found that many of our customer programs did not run properly, and SBC very quickly lost most of their machine rental customers to other companies that still had real 7094s. We also found out that both OS/360 and ASP were very unstable. Today we would probably call this an alpha level system! For example, every time the card reader experienced an error reading a card, the whole ASP system would crash. This happened quite a lot and a system restart took at least 15 minutes. The other thing we found was that the 40 could not really keep up with the 65 when it tried to generate print spool output. This caused the 65 to sit idle a good fraction of the time.

I left SBC in 1967 and dragged my wife and family off to New Jersey to work at Bell Laboratories. Mainly I did this because I felt bad that my mathematics background was not being used enough. Also, things got

a little slow at SBC and I got bored. I found out later that most of the people at my SBC office left not long after I did.

The following figure shows an IBM 360 model 67 computer.

Chapter 5. Bell Labs

After leaving SBC I worked for two years at Bell Telephone Laboratories (Bell Labs) in Holmdel, New Jersey.

The following figure shows me and my family standing in front of Bell Labs, Homdel.

Environment

After leaving SBC, I went to work at AT&T Bell Laboratories in Holmdel, New Jersey. At the time (1967), most computing there was done in Fortran with programs stored on card decks, just as they had been at SBC. My projects included telephone traffic simulation programs and research attempting to correlate the price of AT&T common stock with other economic indicators. I didn't stay at Bell Labs very long, because I found out I liked writing software a lot more than doing research.

The Lab was a research institution, and at Holmdel they tended to treat computer programmers like dirt because programming was considered to be a clerical task. This got them in big trouble while I was there. They had a programmer named Trude who was the only person that understood the giant Fortran program called TELSYN, which was used to simulate telephone switching networks. Because Trude was "only" a programmer, Bell Labs laid her off. Then the researchers in my department discovered that a bug in the Fortran program Trude had maintained caused them to publish several papers with wrong results. None of the researchers knew how to read or fix the code. In a flash they hired Trude back as a consultant to do it.

In 1967, Bell Labs was still collaborating with GE and MIT on an operating system project called MULTICS. This was a very ambitious effort of both computer hardware and software to produce an interactive

computing environment. I never got to use it, but I heard people talking about how great it was going to be “someday.” Someday never came.

The Holmdel lab was in the process of investing in OS/360. The machines I used to run my programs were IBM System/360 40/65 Attached Support Processor (ASP) systems running OS/360. I got several turnarounds a day, but I never even saw the computers. They were kept hidden in rooms in the basement. As an OS/360 user I got to learn about Job Control Language (JCL) and the OS utilities. It amazes me that some people are still wrestling with this stuff 30 years later! JCL was ugly then and it is still ugly now. Calling it a “language” was really stretching things.

While at the Lab I got my first exposure to interactive computing. We had access to a GE Timesharing terminal in my hallway, which was a Teletype machine with a paper tape reader/punch. It was slow and noisy, but having a computer talk to me directly was really a kick. I think this was my first chance to play a game on a computer. It was tic-tac-toe written in BASIC. We weren’t supposed to be using the computer to play games!

People

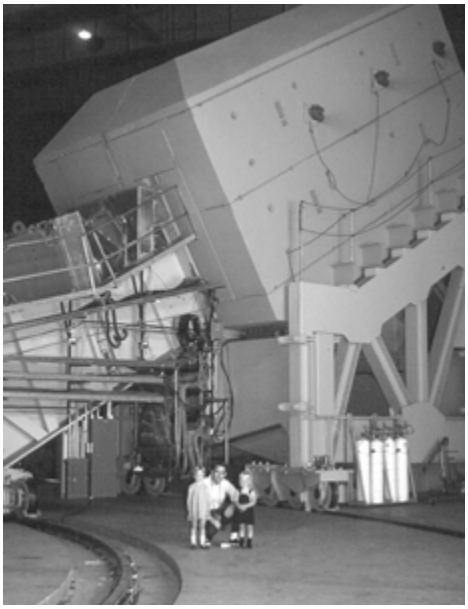
My manager at Bell Labs was Gerald Faulhaber. Under him I carried out my stock market study. As of this writing (2003) Gerry is a professor at the Wharton School at the University of Pennsylvania. At one point in the study, Gerry and I and others went to visit Claude Shannon at his house in Massachusetts. Claude was an MIT professor retired from Bell labs and known to be very knowledgeable about the stock market. His house was full of science toys and practical jokes. Shannon was the father of information theory while at Bell Labs. He also did pioneering work in artificial intelligence.

One of my co-workers was Peete Baer, who left Bell Labs shortly after I did and joined me at SLAC. Peete and I used to play recorder duets on the Bell Labs lawn at lunch time.

Chapter 6. SLAC

The Stanford Linear Accelerator Center (SLAC) is located in Menlo Park, California.

The following figure shows me and my family at SLAC's End Station A during a family day celebration.



The IBM 360/91

I left Bell Labs in 1969 and took a job at the Stanford Linear Accelerator Center (SLAC), a high-energy physics research center in Menlo Park, CA. SLAC was operated by Stanford University under contract to the Atomic Energy Commission (AEC). SLAC did research on the fundamental particles of the universe. The research was done by sending electrons two miles down the accelerator and crashing them into targets so they could observe the shower of collision products.

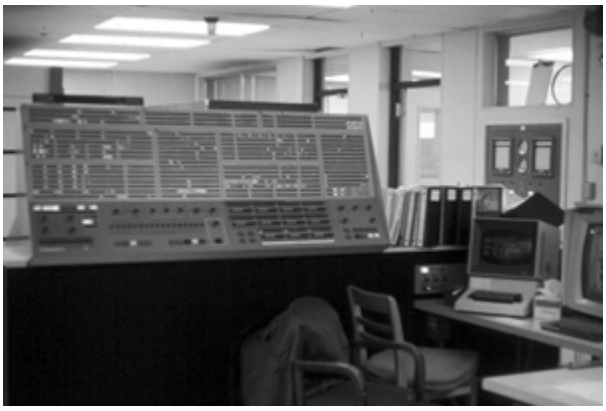
When I got to SLAC they were running physics research computing batch jobs on an IBM 360/91 computer. In 1969, this was one of the most powerful computing machines on Earth. It could run at 3-16 million instructions per second (MIPS), depending on the workload, and had 2MB of core memory. The machine had cost SLAC roughly $8 million. The console of the 360/91 had a flashing light display that was about 3 feet high and about 10 feet long. When the machine ran, the lights changed in all kinds of patterns. Only the IBM customer engineer (CE) knew what the lights meant.

The 360/91 was a very touchy machine. If it got powered off for any reason, it sometimes took days to get it working again. Main memory was divided into eight 256K banks, called BSMs, that were each about the size of a modern refrigerator. IBM kept several spare BSM units around because the memory failed all the

time. Failure was so common that IBM supplied SLAC with two CEs that were on site all the time just to keep the machine running. You can afford to do this if the customer pays $8 million for a computer!

Also attached to the 91 were 2305 drums. These held very little data, but had a head per track, so they were very fast.

The following figure shows the SLAC IBM System/360 Model 91.



In 1969, SLAC was running the HASP OS/MVT operating system on the model 91. Most physics jobs were large (300K) Forgran programs that processed physics experimental data stored on magnetic tape. Many of the SLAC computer users were still storing programs on punched cards, but there were a few 2260 video terminals connected to a software system called CRBE (Conversational Remote Batch Entry) that ran on the model 91. CRBE was a primitive text editor and job submission system, running and storing programs, data and output on disk. Users accessed CRBE terminals in a large room we called a “bull pen.”

The operating system was unstable and we of the Systems Group were forever fixing bugs and installing IBM fixes and our own local enhancements for it. For this purpose we reserved the machine every weekday morning from approximately 7:00 to 8:30 a.m. Not exactly a non-stop operation! (There was notable exception to this regular morning outage. One day in 1974, when I came to work the 91 had been left in production. That was because the Richter physics collaboration had just discovered the charmed quark and were busy verifying their results! (This discovery was good enough to win Dr. Burton Richter the Nobel prize.)

I remember that we had a lot of problems with the Fortran IV compiler. What is funny is that when I left SLAC in 1986, they still had a full-time person installing IBM fixes for the Fortran compiler!

User and system data was stored on banks of 2314 disk drives. Each disk drive held 29MB and a bank held 233MB. At this writing (2003) compact flash cards that fit in a shirt pocket hold 512MB.

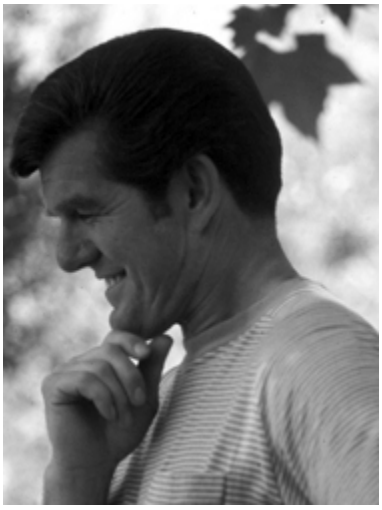
The following figure shows a bank of IBM 2314 disk drives.

There were a few 2250 graphics displays attached to the 91. The 2250s had a screen about as big as a 19-inch TV and were very expensive. Use of the 2250 was severely limited because you needed to tie up 300K of the 2MB of memory to run the batch job driving the display. Sometimes on weekends, the physicists ran a Space War game on the 91 that used the 2250 displays.

The 91 was a real, not virtual, memory machine. SLAC users had to use many tricks to fit large programs into the limited memory using overlay structures. Some of the physics users were very clever at creating those overlays. Using overlays meant the user had to decide when code should be read in to overlay in memory other code that was no longer needed. This was what people did before computers supported virtual memory and paging.

When I first came to SLAC I became manager of User Services. We ran a consulting office where we answered questions and diagnosed system bugs affecting the users. This was my only management job in computing. My manager and mentor was Mel Ray. (Mel went on to work at the World Bank after he left SLAC.) My department included myself, Ted Syrett and Bernie Tice. We later added John Ehrman as a consultant.

The following figure is a picture of Mel Ray.



In addition to managing User Services, I also worked on a program execution profiler called PROGLOOK. After a while I moved into the Systems Group to work on computer measurement and dealt with such things as System Management Facility (SMF) data gathered by OS/360, computer accounting and measurement. Since the hardware was so expensive, we could afford to have people work a lot trying to optimize its use. My manager in the Systems Group was Ted Johnston. I worked for Ted for almost 15 years.

In the early 70s, SLAC moved to a computing environment developed at the main Stanford campus computer center that included:

- MILTEN - software that controlled hundreds of 2741 terminals connected to the 91
- WYLBUR - a line-mode text editor that could be used to create files, submit batch jobs, and look at job output
- ORVYL - an interactive monitor that allowed people to do a simple form of interactive computing by running BASIC programs

John Halperin worked on MILTEN and Joe Wells took care of enhancing WYLBUR/ORVYL. WYLBUR was a big step forward from CRBE, the users liked it a lot, and for many years was in use at universities and other institutions all over the world. It provided more function and used a lot less computing resource than the IBM equivalent product, which was Time Sharing Option (TSO).

About this same time, I bought my first home terminal. This was an ADM3 ASCII terminal kit that we got through the Stanford University computer center at a discount. The kit had over 2000 solder joints and took many hours to complete. I logged on to the SLAC computer from home with the ADM3 using a 300-baud acoustic coupler modem. Now people think 50KB is slow! Primitive as it was, this terminal saved me many trips to SLAC to fix problems.

The following figure is a picture of my daughter Ellen with the ADM3.

Here are some of the people I worked with at SLAC (several more are in the picture gallery at the end of this chapter):

- Ted Johnston. Ted was my boss and the Manager of Systems Programming almost the entire time I worked at SLAC.
- Chuck Dickens. Chuck was director of the computer center.
- Joe Wells. Joe maintained and enhanced SLAC WYLBUR. He later went on to work at the IBM Yorktown Research Center.
- Paul Dantzig. Paul was the son of George Dantzig, who helped invent linear programming, I still have a copy of George's Linear Programming and Extensions on my bookshelf. At this writing (2003), Paul is at the IBM Yorktown Research Center in Yorktown Heights, New York. Paul taught me quite a lot about automotive mechanics.
- Sam Fuller. I knew Sam as a graduate student in the Stanford EE department. He later became Vice President of R&D at DEC.
- Bill Weeks. I hired Bill as a student from Stanford, where he had been a psychology major.
- Joan Winters. Joan was a User Services consultant and our usability expert. She also worked on online help systems and was a leader and participant in many SHARE activities.
- John Ehrman. John came to the computer center from the Computation Group and was very knowledgeable about 360 Assembler language. He was also very active in SHARE. Like me, John later went to work at IBM.
- Others. I once sat in a meeting at SLAC with Linus Pauling, and in other meetings with Donald Knuth and George Forsythe. These people never knew me; I was just sitting in the room. From the

physics world I got to see Pief Panofsky, Burton Richter, and Marty Perl, among others, quite a few times. Since SLAC had so many Ph.D. researchers on its staff, nobody was referred to as “doctor.” We talked about Pief, not Dr. Panofsky. Marty and Burt both won Nobel prizes.

Note:
Even though the Nobel prize was awarded to individual physicists, the work of each physics experiment was done by collaborations with hundreds of individuals in each. Only the top person of the collaboration got the award and the money.

I worked in Richter’s group once for two weeks helping them to convert to the new VM operating system. The group members treated Richter like a god and obviously treasured every word he said to them. During the whole time I was there, Burt never came out of his office to talk to anybody, so I didn’t get to meet him.

The Triplex

Later in the 70s, SLAC did a major computer acquisition (these happened about every 5 years) and bought two IBM 370/168s to augment the 360/91. We called the three computers the Triplex. The 168s each had 3MB of memory and were the first virtual memory machines I ever worked with. The whole Triplex system was controlled by IBM’s ASP MVT and SVS, still with MILTEN/WYLBUR/ORVYL. The 2314 disk drives were upgraded to 3330s and 7-track tape drives were supplanted by 9-track drives.

The following figure shows the Triplex arriving at SLAC.

When all the Triplex hardware arrived, it filled up the computer building we all had been working in, and most of the staff was forced to move to new offices. Those offices ended up being trailers that the AEC had surplused from the Nevada Test Site where atom bomb testing had been done. Many SLAC people hated working in the trailers so much that they quit their job. I hung on and we did eventually get a new

computer center built. Interestingly enough, this was because we failed a security audit for the computers, not because SLAC felt sorry for the people working in trailers.

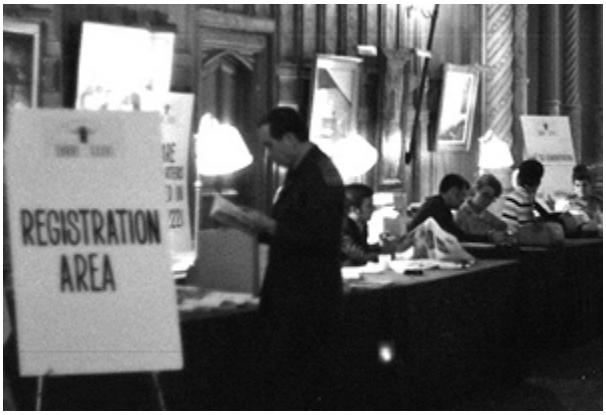
The following figure shows the SLAC computer center trailers.

IBM SHARE

The users of large IBM computers all belonged to a user group called SHARE. (Commercial customers had a similar organization called GUIDE.) SHARE was founded in August 1955. I went to my first SHARE meeting in the late 60s, but SLAC was very active at SHARE, and I went to over 50 SHARE meetings while at SLAC. My manager, Ted Johnston, was a big backer of SHARE attendance for the staff. The SHARE meetings were held in large hotels in big cities like Los Angeles, Anaheim, Houston, Chicago, and New York. For a while I knew my way around many of the big hotels in the U.S. At its peak, SHARE would host over 6000 people at a meeting. The attendees were from all the largest corporations, government organizations, and universities. When IBM was so dominant in the computer industry, prior to about 1985, SHARE was *the* computer meeting to go to.

I gave quite a few talks at the SHARE meetings. This is how I got to be comfortable speaking in front of a large audience. One talk I gave on hardware measurement of an IBM mainframe had several hundred people in attendance.

The following figure shows the registration area at SHARE 71.



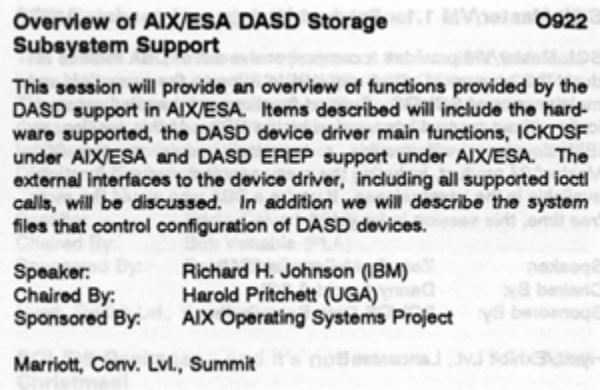
At SHARE, I first worked in the Computer Measurement and Evaluation (CME) Committee, where we discussed how to monitor and instrument the large mainframe computers of the day. Tom Bell from RAND managed the CME group. CME worked with IBM designing an SMF (System Management Facility), which was part of OS/360. Later I moved to being a member of the CMS Project when I became more involved in interactive computing in the 70s and 80s.

The thing some people at SHARE meetings aspired to was to become a SHARE officer. I was never a SHARE officer. Officers got to hear IBM “secrets” that normal SHARE members did not get to hear. Officers wore colored ribbons on their suits and were called ribbon-wearers. I later learned when I went to work at IBM that some of the “secrets” were carefully planted by IBM development groups that wanted to drum up customer excitement over their latest development project. Some of the IBMers were very good at getting customers to say they wanted to buy things that IBM developers wanted to work on.

At the peak era of its computer industry dominance, IBM was very tight-lipped about what it was going to do and it rationed the introduction of technology so as to maximize revenue. There were actually companies that bought first-day order positions for new IBM hardware that they would sell to other companies like shares of stock. The machines always cost about $5 million and came out every 5 years.

After I left SLAC for IBM, I served as the IBM representative to the VM System Management Project (see next chapter). I got a very different perspective on SHARE attending as an IBMer!

The following figure shows the description of a talk I gave at SHARE in Atlanta in 1992.



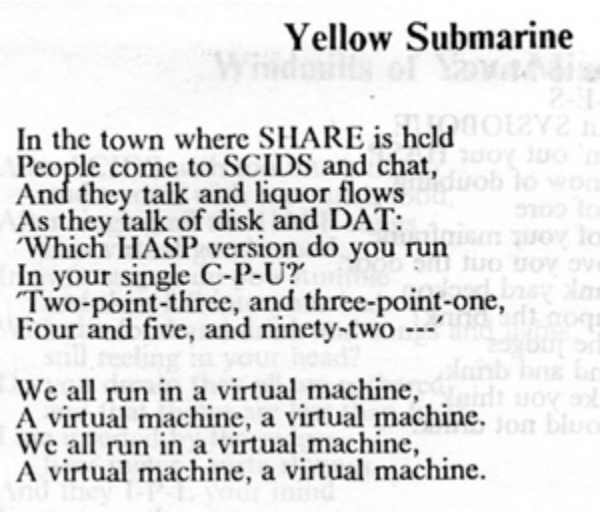
Overview of AIX/ESA DASD Storage Subsystem Support **O922**

This session will provide an overview of functions provided by the DASD support in AIX/ESA. Items described will include the hardware supported, the DASD device driver main functions, ICKDSF under AIX/ESA and DASD EREP support under AIX/ESA. The external interfaces to the device driver, including all supported ioctl calls, will be discussed. In addition we will describe the system files that control configuration of DASD devices.

Speaker: Richard H. Johnson (IBM)
Chaired By: Harold Pritchett (UGA)
Sponsored By: AIX Operating Systems Project

Marriott, Conv. Lvl., Summit

SHARE meetings were intense. I would typically attend 5-6 sessions during the day on various IBM technical topics or project working sessions. At night there were also impromptu meetings called “birds-of-a-feather” that lasted as late as 9:00 p.m. At 6:30 p.m. every night of the meeting there was an open bar in a large ballroom where everyone tended to collect to talk and to make plans for going out to dinner. This was called SCIDS, and I don’t remember what it stands for. On Thursday night at SCIDS around 9:00 p.m. everybody in the room sang silly songs from the HASP (Houston Automatic Spooling Program) songbook. Here is one I sang many times.



Yellow Submarine

In the town where SHARE is held
People come to SCIDS and chat,
And they talk and liquor flows
As they talk of disk and DAT:
'Which HASP version do you run
In your single C-P-U?'
'Two-point-three, and three-point-one,
Four and five, and ninety-two ---'

We all run in a virtual machine,
A virtual machine, a virtual machine.
We all run in a virtual machine,
A virtual machine, a virtual machine.

VM at SLAC

In the late 70s, SLAC was getting ready to move its batch computing from the SVS operating system of IBM to the newer MVS. MVS is the IBM mainframe operating system even today, but it is now called z/OS. As the SLAC Systems Group began work on MVS, the SLAC physicists complained that what they really wanted was interactive computing, not more batch computing. At this time Walt Doherty from the IBM Yorktown Heights Research Center came to SLAC on a visit as a software evangelist and got SLAC

physicists all excited about the Virtual Machine (VM) operating system being used at Yorktown. VM was very strong on interactive capability, but had very little batch function. SLAC decided to go with VM and to ditch MVS. IBM first released VM to customers in 1965 as VM/370. It still exists today, but is now called z/VM.

I became part of a small group leading the conversion to VM. For me this was a great career opportunity, because all the other SLAC system programmers had established turf on the old batch system and were reluctant to move to something new. I went charging ahead working on VM along with just a small group of people, and in less than a year I was a VM expert at SLAC.

SLAC began by running VM on part of one of the Triplex 168s. In 1981, after another mainframe acquisition, SLAC bought a new IBM 3081 computer and ran VM on that. This machine was the first 3081 installed at a customer site and also had the first IBM 3380 disk drives to be installed. We had a steady stream of outside visitors coming to see this new hardware. The 3380s were not yet an announced product, so they were actually chained and padlocked with what we called a “chastity belt” so that nobody could look inside the boxes. The 3081 was the first of a new line of IBM mainframe computers. It was much smaller than the Triplex computers (also much faster), and did not require on-site CEs like the older machines had. So we said good-bye to the several IBMers that had been at SLAC on a daily basis for years. This move saved IBM money, but the company also began to lose touch with its customers because of it, as well.

The following figures show the VM logon sequence.

VM was a completely source-based operating system (written in IBM 370 Assembler) and I had great fun working on modifications to the CMS part of it, which provided the interactive function for the users. I also worked on VM service machines (servers) that performed tasks like disk backup, disk space maintenance, e-mail routing, etc. I worked on this from 1978 until I left SLAC in 1986. My specialty was improving the performance of CMS, since that was what most of the users were using at SLAC. I instrumented CMS file system I/O and found ways to minimize I/O latency by reorganizing files on disk and even added the first primitive read-ahead to the CMS file system. These efforts were the subjects of many talks I gave at IBM SHARE meetings.

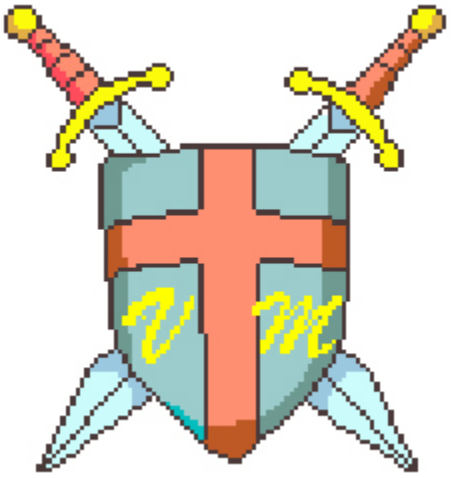
Some of things that were introduced into VM by IBM have since become quite commonplace. These included:

- Electronic mail (e-mail)
- Scripting languages for writing software
- Full-screen editing
- Services in virtual machines

One VM-related project I worked on in the early 1980s got written up in an article in PC Week magazine (“Small Computers and Big Science”). I was able to interface an IBM PC with hardware on our 3081 mainframe called the “Tailgate RPQ.” This allowed the PC to measure the internal activity on the 3081 and display it on a TV monitor. The PC also sent alerts to system programmer terminals when the state of the 3081 did not look normal. I gave another SHARE talk on this work to a large audience, but I never heard of anybody else doing anything like it. This was probably the most enjoyable project I ever worked on.

In August 1987, a year after I left SLAC, I was made one of the honorary “knights” of VM by the SHARE VM Project. I still have my certificate. I was dubbed “Sir Richard, the Innovator.”

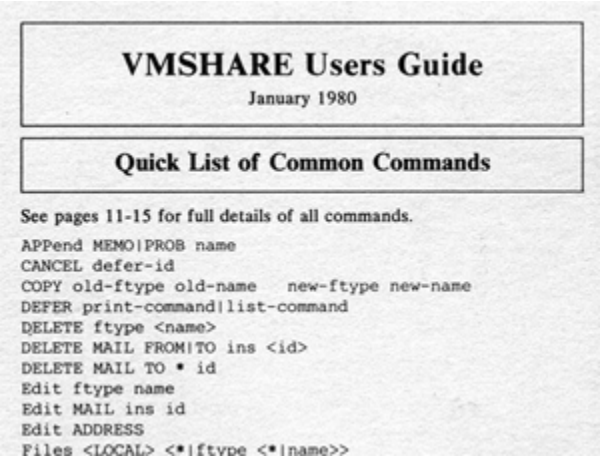
The following figure shows the VM project shield.



BITNET and VMSHARE

Long before the advent of the Internet, around 1981, SLAC was part of a large computer network called BITNET. This network linked thousands of computers all over the world and allowed users to exchange files and e-mail. The network fostered collaboration from customers on improving the utility of VM. Many of us were part of a collaboration called VMSHARE that included a bulletin board system devoted to various topics on improvements to VM. We also used it as a way to exchange information about problems and fixes we had found. IBM distributed all the source code for VM to VM customers, so they could make fixes and improvements of their own very quickly. In the present era a similar thing is happening with the Linux operating system and other open source projects. As the saying goes: this has all happened before and it will happen again.

The following figure shows my VMSHARE reference card.



VMSHARE Users Guide

January 1980

Quick List of Common Commands

See pages 11-15 for full details of all commands.

```
APPend MEMO|PROB name
CANCEL defer-id
COPY old-ftype old-name   new-ftype new-name
DEFER print-command|list-command
DELETE ftype <name>
DELETE MAIL FROM|TO ins <id>
DELETE MAIL TO * id
Edit ftype name
Edit MAIL ins id
Edit ADDRESS
Files <LOCAL> <*|ftype <*|name>>
```

1984, the year of travel

In 1984, the work on VM at SLAC had attracted interest from other physics laboratories in Europe and Japan. SLAC started to get requests for exchanges between our programmers and those of the other labs. As a result I was able to go on some interesting trips that year.

IN2P3 and CERN

In July 1984, I traveled to physics labs in Geneva and Paris as a consultant to help get the SLAC version of VM running at both places. I started at CERN (Organisation Europeanne pour la Recherche Nucleaire) in Geneva. CERN is the largest particle physics research lab in Europe. It is located near Lake Geneva on the border between France and Switzerland. I spent most of my working time at CERN talking to Sverre Yarp and people on his computer system group staff, mostly explaining how the SLAC system worked. After a while it became clear that they were not really going to just install the SLAC software, but were going to rewrite it all to their own needs. CERN had a big budget and could afford to do things like that. In my off hours I got to see lots of the sights in Geneva and did a train tour all around Lake Geneva ending up at Chillon.

The following figure shows the physics research area at CERN.

My next stop after CERN was a French physics lab, called IN2P3 (Institut National de Physique Nucleaire et de Physique des Particules). It was located at the University of Paris. I spent two weeks there helping the IN2P3 Computer Center (Centre de Calcul) install some of the SLAC modifications to the base VM operating system. My main contact there was John O'Neill, a U.S. ex-patriot. I enjoyed working with John and other people on the staff, and I got to wander around Paris on nights after work and one weekend for many long walks. I was truly smitten by the Paris bug and have remembered the place fondly ever since.

The following figure shows IN2P3 and the Notre Dame cathedral.

IBM Japan and KEK

In October of 1984, I went to Tokyo, Japan, to help IBM Japan put together a bid for computing facilities at the KEK physics lab (north of Tokyo). I spent my time explaining to people on the IBM Japan staff how SLAC had used VM for physics work. I found out later that IBM Japan lost out in the bidding process against Fujitsu, who basically gave away their computers to get the bid. I was able to tour Tokyo evenings and on one weekend.

The following figure shows the office where I worked at IBM Japan.

SLAC picture gallery

Chuck Dickens

Ted Johnston

Ilse Vinson

Joan Winters

Bill Weeks

Lee Wilcox

Dave Rossetti and Bob Cook

Bernie Tice and Ted Syrett

Chapter 7. Personal computers

In this chapter I talk about my first experiences with personal computers.

Apple II personal computer

I first got interested in personal computers as a hobby outside of work. When I got my first personal computer (PC), it was more like an addiction than a hobby. I spent countless hours playing with and learning about it! For a person who was only able to work with big computers locked in machine rooms, having one on my desk at home was quite thrilling.

My first PC was an Apple II I bought in 1978 for $1300. That got me a machine with 48K of memory, a BASIC Interpreter and a cassette tape recorder for storing programs. The display was a small color TV set that could show only 40 characters of upper case characters per line and was very hard to read. When my son Tim and I went to buy the Apple II at Computerland of Cupertino, we literally had to wave a checkbook in front of the clerks to get their attention. Sales people at computer stores have been bad from the very beginning, and still are. Keep in mind that in 1978, $1300 was quite a bit of money. Around this time many other people also began buying PCs because they were somewhat affordable.

At first we wrote and played BASIC games on the Apple. The games included Chess, Star Trek, and Adventure. Some were copy-protected, and I used to spend many hours breaking the various protection schemes just for the challenge. It was a popular activity for a while! Now this is called hacking and is considered evil. The Apple II had no printer, no modem, no hard drive, not even a floppy drive, so you could not do anything useful with it. I did learn a lot about PCs and hardware, though. Later we added a $600 floppy drive, with diskettes at $5 each.

The following figure shows our Apple II computer with my son Tim at the controls.

The kids in our neighborhood in Cupertino had a well-organized group effort to copy games for the Apple computer so they wouldn't have to buy each one. Even then the games were expensive. There was a network of mostly teen-aged boys that engaged in this. They were experts at using all the copy protection-breaking programs. They could copy anything!

IBM Personal Computer (PC)

In 1981, we traded the Apple II for one of the first IBM PCs that my wife Anna bought through IBM payroll deduction for over $5000. I hesitate to even calculate how much $5000 in 1981 dollars would buy today! This machine had a 5 MHz Intel 8088 processor, 64K of memory, two diskette drives holding 160KB each, a monochrome monitor, a dot-matrix printer, and a modem. It cost more than the Apple, but it could actually do useful work. Anna and I used software called EasyWriter to write letters to our kids in the summer and a spreadsheet called VisiCalc to do some simple financial calculations. Anna and I even wrote a menu-planning program in BASIC.

The following figure shows the original IBM PC.

In the middle 80s, we traded up to an IBM PS/2 model 60 (80286). Now we had 3MB of memory, a 44MB hard drive and could run Windows and OS/2. It turned out that both the PS/2 architecture and OS/2 were a big waste of time and money for us. Considering the price we paid for it, this machine was a big disappointment. We found out later that with the PS/2, IBM had indeed “snatched defeat from the jaws of victory.” Other companies like Compaq grew rapidly by giving customers the kind of PC they wanted.

In the late 80s, we gave the PS/2 to our daughter Ellen and bought a Compaq Prolinea. This was a 20 megahertz 386 with 8MB of memory and a 120MB hard drive running Windows 3.11. Almost as soon as we bought it we added 8MB more of memory and traded the hard drive for a 540MB drive. PCs have been hungry for memory and disk space ever since.

Since then my family has gone through many desktop and laptop PCs. Too many to list here!

The career benefit I got from spending so much time with personal computers was an interest in hardware, and a confidence that I could work on and understand the guts of computer hardware. This ended up being important in my next job, at IBM.

Chapter 8. IBM

This chapter is about my experiences at IBM.

"n+1 trivial tasks are expected to be accomplished in the same time as n trivial tasks." —Gray's Law of Programming

IBM San Jose

The following figure shows one of my early IBM business cards.

By 1985 it was becoming clear to me that most of the interesting work on VM at SLAC was over for me. IBM had been steadily improving the VM product and I found myself taking out more local SLAC VM modifications every time we got a new release of the product. I also began to get the itch to work somewhere else and see more of the world. This was my version of the mid-life crisis you have all heard about, and have maybe experienced yourself. I decided to try to get a job at IBM, because at the time IBM was the computer company and I knew many people who worked there, having met them at SHARE meetings and the various joint efforts SLAC had done with IBM Research at Yorktown Heights.

In August 1986, I went to work for the IBM Storage Subsystem Division (SSD) in San Jose. I came in as a Senior Programmer (which was a rare event at the time) and entered the VM Technical Office with Margarita Chieng as my manager. In SSD there was a programming laboratory under Lynn Yates, an IBM vice president. The programming lab consisted of almost 1000 people in San Jose and Tucson, Arizona, and was responsible for providing operating system device support for SSD disk and tape hardware. Most of the people worked on MVS device support. I was in a much smaller group under Don Bussey (later Monte Anglin) providing device support for VM. My knowledge of VM was what got me the job.

The following figure shows a picture of the people in the VM Technical Office in 1989.

At the time I got there, SSD designed and built the lion's share of disk and tape storage for all computers. A significant share of IBM's revenues came from selling 3380 disk drives and 3880 disk subsystems built in San Jose. The disk drive assembly line was right there in Building 50 at San Jose. I toured the line soon after I got there. How this picture changed in the years I stayed at IBM! Just as I was about to retire from IBM, IBM sold the remnants of its disk drive business to Hitachi, including the San Jose plant site. In 2001, the disk business actually lost huge amounts of money for IBM.

Our Vice President Lynn Yates had a ranch in Hollister and wore cowboy boots to work. This had established some kind of work apparel norm, and there were quite a few people walking around the halls in my part of SSD wearing cowboy boots. I wore cowboy boots in the 3rd grade, but not at SSD! Yates also had his own version of salty language, which I used to hear quite a bit in the hallways.

When I arrived, the main activity in the Yates organization was developing device support for the MVS operating system (VM was only a sideline.). There were almost 1000 people working on this in San Jose and Tucson, Arizona. They were accustomed to working in a slow and bureaucratic way. At one point they were actually holding pre-pre-plan meetings. The plan meeting was held twice a month and in advance they held meetings to prepare for the meeting to prepare for the meeting! The funny thing was that Yates, our vice president, frequently attended almost all these meetings. Fortunately I had to attend only once or twice. Some managers liked to attend because it got them executive "exposure."

Projects I worked on at IBM

In the next few sections I describe some of the projects I worked in during my tenure at IBM (1986-2002).

VM 3390 (Carmel) device support

My first project when I arrived in August 1986 was to size (estimate) the effort of adding support for the 3390 disk drive (code-named Carmel) to the VM operating system. The 3390 was the successor to the highly profitable 3380 disk drive. The 3390 project had been under way for several years when I arrived (in those days IBM took their time developing products, since they had no real competition). I struggled with this sizing task because I was more used to actually writing the code than just sizing it. I did write a prototype over the 1987 Christmas holidays break that included most of the necessary code.

When I got to IBM their software development process was very slow, complex, and inefficient. We were supposed to produce architecture, high-level design documents, low-level design documents and then finally write and debug the code. This all took far more time, documents and meetings than I thought necessary. You could easily spend six months writing development documents and reviewing them without writing a single line of code. Most of the people I met were used to working this way. Coming from a research environment, the culture shock was considerable! At SLAC we just wrote the code, and almost never did any documentation. I did finally produce the sizing and the necessary development documents and the developers added support for the 3390 to VM.

Note:
at this time SSD had many meetings and task forces going on trying to figure out why the 3380E disk drive was not making its forecast revenues. This was the first disk IBM produced that had that problem. This was a sign of future major problems for the division that they never really overcame.

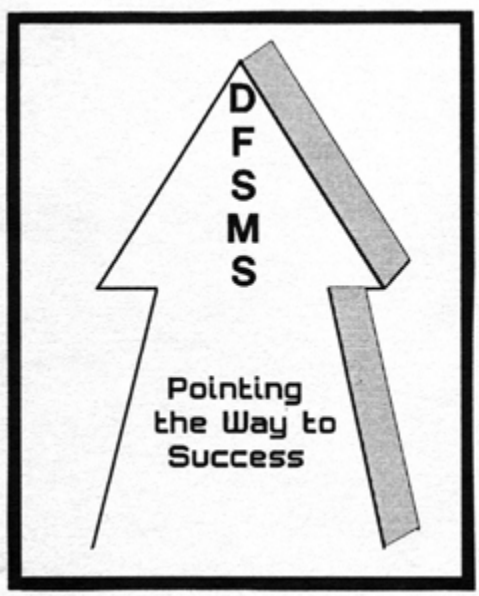
DFSMS/VM datamover

One thing that would be needed if SSD was going to sell the new 3390 disk drive to existing VM customers with older 3380 disk drives, was a way to move existing data between 3380 to 3390. The geometry of the 3390 was different from that of 3380. It had longer tracks than 3380s. Without telling my manager, I began to work on a prototype of a utility that could move VM/CMS file systems (on minidisks) between 3380 and 3390.

At this same time, SSD had a large number of technical people from San Jose, Tucson, and the East Coast who were architecting more powerful storage management for all the IBM operating systems under the code name Jupiter. Once they find out about my prototype datamover utility, I was put to work in software development on a product called DFSMS/VM. I can't remember what all those letters stood for! This was supposed to be stage one of the Jupiter Project for VM.

The product combined the existing ISMF user interface from MVS with my datamover and server virtual machines to make it possible for customers to easily move their 3380 data to 3390. Mike Dillon led this product development and I supplied the mcopy datamover. Working on the datamover (mcopy) taught me a lot about the details of I/O programming for IBM disk drives. Mcopy used complex channel programs that moved data between drives as fast as the drives could go.

The following figure shows an example of DFSMS advertising.



This project gave me my first introduction to IBM divisional rivalries. The VM operating system was developed in a different division of IBM in Endicott and Kingston, NY. Once the people in that division found out what we doing in San Jose, they didn't like it and tried to stop us from shipping our product. This was the standard mode of operation inside IBM at the time. We were able to ship the product because there was too much potential hardware revenue involved. There was a saying I learned from this: "Never get in front of the IBM revenue train!".

The product was used by many large IBM customers to move huge amounts of data, and we never had a single data integrity problem reported. I was proud of this. IBM did not make much money from selling DFSMS/VM. It was given to most large customers as part of a 3390 disk purchase.

In SSD there was always a tendency for the hardware people to want to give away software to make it easier to sell the hardware. The argument was that the hardware revenues completely dwarfed anything that could be produced by selling software. To most of SSD management, software revenue was uninteresting. Later on we all learned that software profit margins were generally much larger than those for hardware and that selling software might actually be a good idea. When I retired from IBM, selling storage software and firmware was our main focus.

AIX/ESA disk and tape support

In the late 1980s, IBM was part of an industry group, the Open Software Foundation (OSF). Part of the 370 processor division of IBM, in Kingston, NY, had decided to develop a version of OSF/1 UNIX to run on the 370 mainframe. This would allow IBM to compete with a fairly successful UNIX offering from Amdahl, which was called UTS. My department, managed by Nora Denzel (now an HP vice president), got the mission of providing device support for SSD hardware in this product, which was called AIX/ESA.

Our team was responsible for the disk and tape device drivers for AIX/ESA. There were about a dozen of us in San Jose working remotely with a much larger group of several hundred in Kinston, NY. I was the technical leader of the San Jose group and was responsible for the design and overall technical direction of the device driver development. Later when the disk driver got into trouble, I became a coder responsible for fixing many bugs and actually getting us through system test of AIX/ESA.

I didn't know it at the time, but this device driver was larger and more complex than most UNIX device drivers ever done before. That was because the hardware we supported from SSD was very complex and required incredible amounts of error recovery code. While I was on this project I made many trips to Kinston during a period of several years. It all taught me a lot.

From working on this project I learned about team leading, negotiating, UNIX, the C language, and many other things. From my standpoint it was a big success. Unfortunately, the product did not sell well once it was produced in 1989-90. We never got more than 35 paying customers. I was amazed to find out that several hundred people had worked on something for several years only to find out that the customers didn't like the product and IBM Sales had not much interest in selling it. In addition, part way through the development of AIX/ESA, the MVS people in Poughkeepsie began working on a rival product called "Open MVS" that allowed existing MVS customers to run some UNIX applications under MVS. This was an example of the corporate immune system at work.

Note:
It is interesting that in all my years at SLAC, our budgets were always managed carefully. SLAC wasted very little money. In IBM I saw hundreds of millions of dollars wasted on just some of the products I worked on. So be careful when people start talking about how the "government" wastes money. Industry can be pretty good at it too!

The following figure shows a picture of Nora Denzel.

Nora Denzel, Senior Vice President Software Global Business Unit, Hewlett-Packard Company

Workplace OS tape support

My team was quickly pulled off of work on AIX/ESA when the failure of the product caused us to lose our funding. About this time, IBM had a large product development effort starting up called Workplace OS. As a way to save money on supporting the multiple IBM operating systems, a micro-kernel was going to be developed that would allow multiple operating system personalities to run on top of the micro-kernel (based on the Mach kernel from Carnegie Mellon University). The micro-kernel would provide the low level hardware support for all the personalities.

Three of us in San Jose were asked to write the SSD tape device support for this product. Primary development was in Boca Raton, Florida, which had been the home of the PC OS/2 operating system. For almost a year our team tried to work with the people in Florida. I traveled to Boca Raton several times for meetings. After a while the Boca folks stopped communicating with us. We learned later that this was because Workplace OS was in big trouble. Enough of the micro-kernel code had been written to demonstrate that performance of the operating system was terrible, and nobody could think of a way to fix it. This caused the inevitable schedule slips and the search for the guilty began. By this time (around 1992) IBM itself was in big trouble and was losing money like crazy. Most of the Workplace OS developers got laid off after Lou Gerstner began running IBM in 1993.

The Mach kernel did have success in the marketplace. Apple Computer used it as the basis for their OS 10 operating system for the Macintosh. Today it is also being used by the GNU HURD developers. In IBM, Mach died a horrible death.

The following figure shows my office at IBM.

ADSM (ADSTAR Distributed Storage Management) device support

After Workplace OS I became part of a new project to provide SCSI device support for a network backup product called ADSTAR Distributed Storage Manager (ADSM). The name ADSTAR came from the name IBM SSD was supposed to have once it got spun off as a separate business under our new division president, former U.S. Senator Ed Zschau. The spin-off never happened, and Zschau left IBM. In 1993, Louis Gerstner, the new CEO, decided to keep IBM together. At this writing (2003), ADSM still exists, but is now called Tivoli Storage Manager (TSM).

ADSM needed to be able to use tape and optical libraries to back up client data. There were many companies providing such hardware products, not just IBM SSD. After some false starts, my team wrote three device drivers using common software building blocks I helped design. We were able to support hundreds of IBM and non-IBM devices with this code. This was critical to the success of ADSM, since our competitors (the main one being Legato Networker) already could support that many devices.

Just about the time we got the code finished, most of the development team left IBM (1993). This was because there were plenty of jobs in the industry and IBM had to lay people off to cut expenses. Our developers and managers found out they could make a lot more money outside IBM, so they left. Some people got 50% raises!

For whatever reason, I hung on, but for a while we had only a few developers left. The nice thing, though, was that unlike my previous products, ADSM was a roaring success. Over several years we built the team back up again with new people and I won awards and filed patents for the work we had done. My biggest ADSM achievement was receiving an IBM Corporate Award from Lou Gerstner himself.

The following figure shows a picture of my ADSM department (me on left).

SSA disk Windows device driver

By 1997, Lynn Yates had been given responsibility for his existing MVS software plus all hardware firmware in the storage division. During the development of the new Tarpon/Shark disk storage subsystem, an SSD group in Havant, England, had greatly embarrassed Yates by missing critical dates that had been promised to the rest of IBM. To fix this, Yates decided that he wanted to move this firmware and associated device driver development work from England to San Jose. The hardware involved was the Serial Storage Architecture (SSA) disk and its controller cards. These disks and controllers were generating lots of revenue at the time for IBM.

I was put in place as the leader of a small San Jose team that was supposed to take ownership of the Windows device drivers for the SSA controller cards. Our team spent over a year learning how the existing card firmware and Windows device drivers worked. This was all done with only the slightest help from IBM SSD in Hursley (the Havant group had moved there). Like most human beings, the Hursley group was not interested in putting themselves out of business by giving their mission to people in San Jose. This resistance could have been overcome by executive management, but by the end of 1997, Yates was on his way out of IBM (he retired from IBM rather suddenly at the end of the year) and we had no executive backing. The Hursley people knew about this, dug in their heels (at one point it took us in San Jose over 6 weeks to get a look at their code), and waited for us to go away, which we did by early 1998. This year and a half of wasted effort was the most frustrating time I had at IBM.

Tarpon/shark multi-pathing

Fortunately, in mid-1998 a more interesting project came along. The Storage Division had decided to get back into the disk subsystem business (which it had almost completely lost to EMC) by building disk subsystems out of common parts. The first such subsystem was named Tarpon and the next was Shark.

Since these boxes contained huge amounts of customer data, the customers wanted to have redundant hardware paths connecting the subsystem to each server. Without this capability, IBM could not compete with EMC, who already had multiple paths in their hardware and software. My new team was given the job of providing device driver software that would support this environment in the AIX, Windows, Solaris, and HP-UX environments. The team initially consisted of me, Limei Shaw, and Cam-Thuy Do. Our manager was Randy George. We called the product Riptide. We all worked very hard for several months and got the product working on Windows and AIX. This time IBM Austin (owner of the AIX operating system) tried to kill our product, but we managed to placate them by making the changes to our AIX code they said they wanted. They weren't able to kill the product because, again, there was too much IBM revenue at stake. It was a good thing, too, because it was not until 2003 before AIX had similar support in the operating system.

This was the last project I worked on before I retired from IBM. By the time I left our team had grown to over twenty people and had produced many releases of the product, which we called either Subsystem Device Driver (SDD) or Data Path Optimizer (DPO). DPO was OEMed to other storage vendors.

Since the Shark program was successful for IBM, and because our multi-pathing software was delivered on time and was very reliable, I was promoted to Senior Technical Staff Member (STSM) about a year before I retired.

IBM business travel and boondoggles

Prior to the corporate near-death of IBM in the early 90s, budgets for business travel were very generous. During these years, especially, I went to many IBM locations to attend meetings. Among the places I visited in the U.S. were:

- Tucson, Arizona
- Endicott, New York
- Kingston, New York
- Poughkeepsie, New York
- Boca Raton, Florida
- Cary, North Carolina
- Hawthorne, New York (also Yorktown Heights)
- Austin, Texas

My most frequent destination was Tucson, where roughly half of the people in my division were located. In addition, I traveled to Kingston and Endicott many times. I was able to gather information for my family history on the trips to Endicott, since I flew out of Syracuse airport which is close to Auburn where the Moseleys came from.

I remember one meeting in Kingston, NY, that was attended by a dozen people from San Jose and Tucson. During this time IBM San Jose and IBM Tucson were constantly battling for turf and hence both sites sent a large complement of people to protect their interests in all the meetings on the East Coast.

In addition to frequent regular business travel, I also had the pleasure of traveling on boondoggle trips, one to Thornwood, New York, for a one-week seminar on the future of operating systems and another one-week trip to Mayaguez, Puerto Rico, to help IBM recruit engineering students. In Thornwood we had rooms with built-in computer offices having access to the IBM corporate network. This looked nice at first, but who wants to work day and night? The predictions I heard on operating system futures were useless. Nobody took the challenge from Microsoft seriously at all. Most of the predictions for the future were wrong.

In Puerto Rico my “job” was to give two guest lectures to computer science and engineering classes at the university. So for those two hours of work I got to spend a week touring all around the island. I must confess it was fun. You can get good fish dinners in Puerto Rico!

IBM Research

Although I never worked for IBM Research Division, I did travel to their locations many times, either on airplane trips to Yorktown Heights and Hawthorne or driving up the hill to Almaden Research in San Jose. Until the middle 1990s, IBM Research was the inner sanctum of computer hardware and software research. All their locations had beautiful buildings, and the staff members generally had much nicer equipment than we lowly "development" people. Three of my co-workers from SLAC ended up at Yorktown Heights: Joe Wells, Paul Danzig, and Mike Penner. The early stages of ADSM product development were a joint project between Almaden Research and SSD San Jose and Tucson.

Awards and patents

IBM has awards programs for technical people that recognizes technical achievement. Usually they give these awards to people who played a significant role in a product that has some level of commercial success. I received quite a few of these while I worked at IBM, and used the money to take some nice trips to places like the Soviet Union and Costa Rica. If you want to see one of the patents, look up 5450579 issued 9-12-1995, Method and Apparatus for Error Recovery.

The biggest prize for me was an IBM Corporate award for my work on ADSM. These awards are given out once a year at a big event where IBM pays for your stay in a big city for a whole week. For me the city was San Francisco, which wasn’t too exotic. During the event I got to meet quite a few IBM executives and also attend an awards ceremony with my wife.

The following figure shows a picture of me and our team receiving our award from the IBM CEO, Louis Gerstner.

The Internet bubble

By 1998, it was clear that an incredible boom was happening in high-tech. The newspapers were full of articles about people going to work at Internet startups and becoming a millionaire months later when the company “went public.” Many people all over Silicon Valley were working 60+ hours a week so their company could ship its first product and go public. My wife and I used to hear people talking all the time (even in Starbucks) about stock options, how much money they had, and whether they could retire by the time they were 30. Many people left IBM at this time to go get rich. IBM itself even began to give stock options to non-executives. I got some myself. One lesson Anna and I learned from the bubble times is that the money from stock options is not real until you spend it. Strangely enough, many people in the high-tech industry held on to their stock options instead of cashing them in, and watched them become worthless instead.

IBM was cautious and did not participate very much in the bubble. During the boom, Sun Microsystems and EMC took market share away from IBM at an alarming rate. When the bubble burst Sun and EMC suffered considerably, but IBM did not at first. However, in the last year I was at IBM they finally did see rapidly falling revenues in various areas of technology, including storage. In early 2002 almost 30% of the people working on Shark were laid off and the entire development effort was moved to Tucson from San Jose. This looked like the handwriting on the wall to me for the location in San Jose where I worked.

My retirement from IBM

I retired from IBM at the end of July 2002. My last year at IBM was during the largest business downturn I have ever experienced. The computer industry went from being white-hot in 1998-1999 to a wasteland by 2001. I had very little to keep me busy at IBM. We had no travel money, and no money for classes or books. Instead of starting new product development, the executives focused mostly on trimming the cost of the things they already had going. This was a good time for me to make my departure, while my pension was still intact!

Chapter 9. Code samples

In this chapter I have included fragments of code I actually wrote myself on various projects.

Programming languages

During all the time I worked in computing, the main thing I was working to produce was code in a computer language. Some of languages are still in use, and some are now close to being dead languages. Languages I have learned and used include:

- Adobe FLEX
- Apple II BASIC
- Apple 6502 Assembler
- C
- DITA XML
- EXEC2
- Fortran H
- Fortran II
- Fortran IV
- IBM 1401 Autocoder
- IBM 7094 Assembler
- IBM PLAS
- IBM System 360/370 Assembler
- Intel 8088 Assembler
- JAVA
- PHP
- Python
- PL/I
- UNIX Shell
- VM EXEC
- VM REXX
- WYLBUR EXEC
- XHTML
- XSLT

ADSM device driver (IBM, C)

```
899     /*
900      * Call the table lookup routine to find the proper recovery action
901      * for the sense bytes from the error.
902      *
903      * Note:  Occasionally, it is necessary to use the last executed SCSI
904      * command in addition to the sense bytes when determining which
905      * action to perform for an error.  This would be the case for devices
906      * which require that different actions be taken for the same sense
907      * code combination when received in response to different SCSI
908      * commands.  For this reason, the operation code of the last
909      * executed SCSI command will be passed to erpfind.
910      */
911
912     recovery_action = erpfind(sense_bytes, recov_table_p, table_width,
913                               scsi_command);
914
915
916     /*
917      * Check that the table lookup found a recovery action.
918      * (If one is not found, there is something wrong with the table!)
919      */
920
921     if (recovery_action == NO_ACTION) {
922
923       erp_rc               = ERP_LOGIC_ERROR;
924       scsiout_p->compcode = DD_DEVICE_DRIVER_FAILURE;
925
926     } else {
927
928       /*
929        * Call the error recovery action processor to carry out the
930        * recovery action called for in the previous table lookup.
931        */
932
933       erp_rc = erpact(recovery_action, scsi_device_type, scsi_command,
934                       dev_p,      helper,           scsiout_p);
935     }
936
937     /*
938      * Return to caller with proper return code set.
939      */
940
941     DD_EXIT_SRC("erp_main: ", erp_rc);
942     return erp_rc;
943
944   } /* end erp_main() */
945
```

VM/CMS internal trace (SLAC, IBM 370 Assembler)

```
*
* MAIN ENTRY POINT WHEN WE ARE INVOKED VIA SVC 202.
* WE COULD BE RUNNING AS A MODULE OR AS A NUCLEUS
* EXTENSION.
*
BASEADDR LR    R12,R15             ADDRESSING STUFF
         LA    R13,2048(,R12)
         LA    R13,2048(,R13)
         LA    R11,2048(,R13)
         LA    R11,2048(,R11)
         USING BASEADDR,R12,R13,R11
         USING NUCON,R0
         ST    R14,REG14           SAVE RETURN
         LA    R1,8(,R1)           ANY OPERANDS?
         CLC   0(8,R1),=CL8'XVZZV' GOT MAGIC WORD?
         BE    GOFORIT             PROCEED IF SO
         CLC   0(8,R1),=CL8'RESET' NUCX GOODBYE CALL?
         BE    NUCXDROP            PROCEED IF SO
IGNORE   LA    R15,100             IGNORE ANYTHING ELSE
         BR    R14
         EJECT
*
*        NUCXDROP IS KILLING US, UNHOOK AND GO AWAY.
*
NUCXDROP DS    0H
*
         ENABLE INTTYPE=NONE       DISABLE INT
         MVC   SUCNPSW(8),OLDSPSW PUT BACK PSW
         MVC   EXTNPSW(8),OLDEPSW PUT BACK PSW
         MVC   IONPSW(8),OLDIPSW PUT BACK PSW
         MVC   ADMSITI(4),IONPSW+4 DMSITE CHECKS THIS FOR BLIP
         MVC   PGMNPSW(8),OLDPPSW PUT BACK PSW
         CLI   BALRFLAG,X'00'      TRACING BALRS?
         BNE   NUCXDRP2            NO, SKIP NEXT STUFF
*
         MVC   ARDBUF(4),OLDRDBUF PUT BACK POINTER TO RDBUF
         MVC   AWRBUF(4),OLDWRBUF PUT BACK POINTER TO WRBUF
         MVC   ARDTK(4),OLDDIOR   PUT BACK DIOR POINTER
         MVC   AWRTK(4),OLDDIOW   PUT BACK DIOW POINTER
         MVC   AUPDISK(4),OLDAUD  PUT BACK AUD POINTER
         MVC   AESTATE(4),OLDSTATE PUT BACK STATE POINTER
         MVC   ADMSFREB(4),OLDFREB PUT BACK FREB POINTER
*
         L     R15,NUCCSLBK        POINT TO DMSCSLBK IN NUCON
         USING CSLBK,R15           MAP IT
         MVC   CSLACCR(4),OLDCSL   PUT BACK CSL POINTER
         DROP  R15
*
NUCXDRP2 ENABLE INTTYPE=ALL        REENABLE EVERYTHING
         SR    R4,R4               ZERO WORK AREA POINTER
         ST    R4,USERLVL          USED BY CTRACE
         CLI   SPOOLFLG,X'00'      SPOOLING ACTIVE?
         BNE   NUCXDRP3            NO, SKIP NEXT COMMANDS
         L     R4,NCCW             ANYTHING IN BUFFER STILL?
         LTR   R4,R4               ...
         BZ    NUCXCPRT            NO, JUST CLOSE SPOOL FILE
* EMPTY OUT SPOOL BUFFER BEFORE CLOSING SPOOL FILE
         S     R4,=F'1'            CCW COUNT-1
         MH    R4,=H'8'            TIME LENGTH OF A CCW
         LA    R5,PRTCCW           ADDRESS OF START OF CHAIN
         AR    R5,R4               ==>FLAG IN LAST CCW
         MVI   4(R5),X'20'         TURN OFF CHAINING
*
```

Multi-pathing device driver (IBM, C)

```
135         /* We have multiple paths to chose from, use appropriate algorithm
136            to pick one. */
137
138         /* cannot used load sensitive algorithm if no counters exist */
139         path_algorithm = dev_p->policy;
140         if ( (path_algorithm == DD_POLICY_OPTIMIZED) &&
141              !counter_flag )
142           path_algorithm = DD_POLICY_ROUND_ROBIN;
143
144         switch ( path_algorithm )
145         {
146           case DD_POLICY_ROUND_ROBIN:
147
148           path_alive      = DD_PATH_NONE;
149           path_alive_same = DD_PATH_NONE;
150           nnl = 0;
151
152           /*
153            * Scan the paths looking for various types of paths for this device:
154            *
155            *  1. A path other than the last one used for this device.
156            *  2. A path for this device that is usable.
157            *
158            */
159
160           for(i=0 ; i<dev_p->path_count; i++)
161           {
162             /* optional detailed trace for grouping */
163             if( dev_p->controller_flag )
164             {
165               DU_DEBUG_S4("path %d defcntrl %d defcnt %d pathcntrl %d",
166                 i, dev_p->default_controller, dev_p->default_path_count,
167                 dev_p->path[i].controller_number);
168             }
169             /* never touch closed path */
170             if(( dev_p->path[i].state == DD_PATH_CLOSED ) ||
171                ( dev_p->path[i].state == DD_PATH_INVALID ))
172               continue;
173
174             if( path_alive == DD_PATH_NONE )
175             {
176               /* In case ALL paths are currently being bypassed, we will
177                  use this one anyway. It is at least not dead yet.   */
178               if( dev_p->path[i].state == DD_PATH_OPEN )
179               {
180                 /* possible path if no suitable path found later */
181                 path_alive = i;
```

AIX/ESA disk device driver (IBM, C)

```
232         /* Also, we only allow I/O in multiples of disk block size */
233         if (bp->b_bcount & (DEV_BSIZE - 1)) {
234                 debug(CKDDBG_B,("ckdstrategy: bcount not multiple of DEV_BSIZE.\n"));
235                 bp->b_error = EINVAL;
236                 bp->b_flags |= B_ERROR;
237                 bp->b_resid = bp->b_bcount;
238                 goto unlock_error_exit;
239         }
240         /* Make sure the 1st block is inside the partition */
241         if (bp->b_blkno < 0 || bp->b_blkno >= pp->p_size )
242                 {
243
244                 /*
245                  *  bio readahead always tries to read one extra block
246                  */
247                 if ((bp->b_blkno >= pp->p_size)
248                         && ((bp->b_flags & B_READ) == B_READ)) {
249                         debug(CKDDBG_B, ("ckdstrategy: blk out of range(2)\n"));
250                         bp->b_resid = bp->b_bcount;
251                         goto unlock_error_exit;
252                 }
253                 debug(CKDDBG_B, ("ckdstrategy: blk out of range(1)\n"));
254                 bp->b_resid = bp->b_bcount;
255                 bp->b_error = EINVAL;
256                 bp->b_flags |= B_ERROR;
257                 goto unlock_error_exit;
258         }
259         /* If the ending block extends beyond the end of the partition,
260            truncate back to the end of the partition. */
261         if (bp->b_blkno + nblks > pp->p_size) {
262                 debug(CKDDBG_B, ("ckdstrategy: bcount truncated.\n"));
263                 bp->b_bcount = (pp->p_size - bp->b_blkno)<<DEV_BSHIFT;
264         }
265
266         /* Set b_cylin (b_resid) for disksort */
267         bp->b_cylin = bp->b_blkoff / fi->fi_disklabel.d_secpercyl;
268
269         /* queue lock fast path */
270          fi->fi_stats.fi_qlock_strategy++; /* count fast path */
271          disksort(&fi->fi_tab,bp); /* sort the requests  */
272          ufcnt++;
```

PC DOS trace (SLAC, Intel 8088 Assembler)

```
341  CODE_START:
342  ; BASIC INITIALIZATION STUFF
343             MOV     AX,CS           ; FIX UP DS
344             MOV     DS,AX           ;
345             MOV     SS,AX           ; AND SS ALSO
346             LEA     AX,BEGSTACK     ; SET SP
347             MOV     SP,AX           ; ...
348  ;
349             MOV     AL,OUTINT       ; TRACING ALREADY ACTIVE?
350             MOV     AH,35H          ; GET CURRENT VECTOR
351             INT     DOSINT
352             MOV     AX,ES           ; POINT ANYWHERE?
353             CMP     AX,0
354             JE      TSTART
355             CMP     BX,0
356             JE      TSTART
357  ;
358             LEA     DX,SIGNOFF      ; TYPE ABORT MESSAGE
359             MOV     AH,9
360             INT     DOSINT
361             MOV     AL,16           ; SET NON-ZERO RC
362             MOV     AH,4CH          ; TERMINATE PROCESS
363             INT     DOSINT
364  ;
365  TSTART:    LEA     AX,BEGTRACE     ; INIT TRACE TABLE POINTER
366             MOV     POINT,AX
367             MOV     AL,YES          ; TURN ON TRACING
368             MOV     TFLAG,AL
369             MOV     AL,NO           ; NOT WRAPPED YET
370             MOV     WRAP,AL
371  ;
372             MOV     AX,CS           ; CONVERT LOAD ADDR FOR PRINTING
373             LEA     SI,LOADCS
374             CALL    HEXPRT
375             MOV     AX,100H
376             LEA     SI,LOADIP
377             CALL    HEXPRT
378  ;
379             LEA     DX,SIGNON       ; TYPE SIGNON MESSAGE
380             MOV     AH,9
381             INT     DOSINT
382  ;
383  ; SET INT FOR PRODUCING OUTPUT
384  ;
385             LEA     DX,OUTPUT       ; ADDRESS OF ROUTINE
386             MOV     AL,OUTINT       ; INT NUMBER
387             MOV     AH,25H          ; SET INTERRUPT VECTOR
388             INT     DOSINT          ; HAVE DOS PLACE IT
389  ;
390  ; Set INT intercepts
```

VM/CMS profile exec (SLAC, IBM REXX)

```
/*
  PROFILE - DJOHNSON PROFILE EXEC.

  REXX version, 10/01/84
                                  */
trace 'o';Address command;yes='Y';no='N'

/*============== Query   Environment ==================*/

'DISKADRM S'  /* IBMCMS? */
pull . . . slabel .
 if slabel='IBMCMS' then 'EXEC SLACINIT'

'CPSTACK QUERY CPUID'   /* Display where I am running */
pull . . cpuid .;machine=substr(cpuid,9,4)

say "You are on the" machine

batch='NO';'CMSINFO BATCH';if rc>0 then batch='YES'
if batch='YES' then say "PROFILE running in batch."

'CPSTACK QUERY CPLEVEL' /* Display time/date of last IPL */
pull .;pull .;pull ipldata
say ipldata

'QUERY CMSLEVEL(STACK LIFO' /*what level of CMS?*/
pull . . cmslevel ',' .
say "Running in SP"cmslevel

'CP .VAR Q LOGON'  /* Is this an IPL or a LOGON */
if rc=0 then logged=yes; else logged=no

if logged=no then do /* set my logon variable for next time */
                  'CP .VAR SET LOGON Logon:' date() 'at' time()
                  end

'QCONSOLE';pull . con . /* get console state */

/*============== Set up  Environment ==================*/

'CP .SET ACNT OFF';'CP .SET EMSG ON'

'EXEC MYDISK' /* set disk config */

'DISKADRM P'  /* got SYSTEMS? */
if rc=0 then pull junk
 else do /* put it there */
      'SET CMSTYPE HT'
      'EXEC GIME SYSTEMS (QUIET REP'
      'SET CMSTYPE RT'
      end
```

VM/CMS XEDIT macro (SLAC, IBM EXEC2)

```
&TRACE OFF
&BUFFER *
&FOUNDIN = &STRING OF Found in:
PRESERVE
SET WRAP OFF
&CASE M
&TARG = &ARGSTRING
SET MSGMODE OFF
&FOUND = 0
TRANSFER SIZE LINE
&READ VARS &LASTL &WASAT
&IF &N LT 1 &GOTO -TELL
&IF .&1 EQ .? &GOTO -TELL
TOP
 &LOOP -LOOP *
CL /&TARG
&IF &RC NE 0 &GOTO -DONE
&FOUND = &FOUND + 1
TRANSFER LINE
&READ VARS &L
&FOUNDIN2 = &CONCAT OF &FOUNDIN &L &BLANK
&LENFOUND = &LENGTH OF &FOUNDIN2
&IF &LENFOUND LT 80 &FOUNDIN = &FOUNDIN2
&WL = &CONCAT OF &L
&WL = &CONCAT OF 00000 &WL
&WL = &RIGHT OF &WL 6
STACK
&READ STRING &DATA
&TYPE &WL &DATA
CL *
 -LOOP
-DONE
&IF &FOUND GT 0 &GOTO -WASFOUND
&ESTR = &CONCAT OF ' &TARG '
SET MSGMODE ON
EMSG &ESTR not found in file.
&IF &WASAT = 0 TOP
&IF &WASAT > 0 :&WASAT
-WASFOUND
RESTORE
&IF &FOUND GT 0 EMSG &FOUNDIN
&EXIT
-TELL
SET MSGMODE ON
&TYPE Format: DJOTYPE string
&TYPE
&TYPE Type all lines containing "string"
&TYPE
&EXIT 100
```

DITA XML (VR Communications, XML)

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE concept PUBLIC "-//OASIS//DTD DITA Concept//EN" "../dtd/concept.dtd">
<concept xml:lang="en-us" id="code_ditaxml">
    <title>VM/CMS ditaxml macro (SLAC, IBM EXEC2)</title>
    <shortdesc id="shortdesc"/>
    <prolog>
        <author type="creator">Richard Johnson</author>
        <author type="contributor">Anna van Raaphorst</author>
        <publisher>VR Communications, Inc.</publisher>
        <copyright>
            <copyryear year="2003"/>
            <copyrholder>VR Communications, Inc.</copyrholder>
        </copyright>
        <critdates>
            <created date="2012-Mar-26"/>
            <revised modified="2012-Mar-26"/>
        </critdates>
        <metadata>
            <keywords>
                <keyword>code_sample</keyword>
                <keyword>DITA XML</keyword>
                <indexterm>code samples
                <indexterm>DITA XML</indexterm>
                </indexterm>
                <indexterm>DITA XML code sample</indexterm>
            </keywords>
        </metadata>
    </prolog>
    <conbody>
        <section>
            <p>
                <image href="../images/code_ditaxml.jpg"/>
            </p>
        </section>
    </conbody>
</concept>
```

Python code (VR Communications - Pearson, Python)

new_password_advanced.py

```
for i in range(tries):
    #
    # get the user's input
    #
    # prompt for the new password
    password = str(input("Enter new password: "))

    #
    # test the conditions and report errors
    #
    # test the password length
    if len(password) >= minlength:
        # test for all alphanumeric
        if password.isalnum():
            # make sure  there is at least one letter
            if not password.isdigit():
                # make sure there is at least one number
                if not password.isalpha():
                    # password passes all tests, mark it valid
                    valid = True
                    # if the password is valid after 1 or 2 tries
                    # break out of this section
                    # and proceed to the "cleanup" section
                    break
                else:
                    # there must be at least one digit in the password
                    print("Error, password does not contain a numeric digit.")
            else:
                # there must be at least one letter in the password
                print("Error, password does not contain a letter.")
        else:
            # password contains an invalid character
            print("Error, password is not alphanumeric.")
    else:
        # password was not long enough
        print("Error, password is less than",minlength,"characters.")
```

Chapter 10. After IBM: Pillar Data System and VR Communications, Inc.

It turns out my *"retirement"* from IBM was only a temporary intermission from high-tech pursuits. Anna and I moved to El Dorado Hills and began to look for things to supplement our leisure-time activities like playing golf or traveling. What follows is a description of the time I spent working at Pillar Data Systems in San Jose, and my ongoing work with Anna for our company, VR Communications, Inc.

Pillar Data Systems

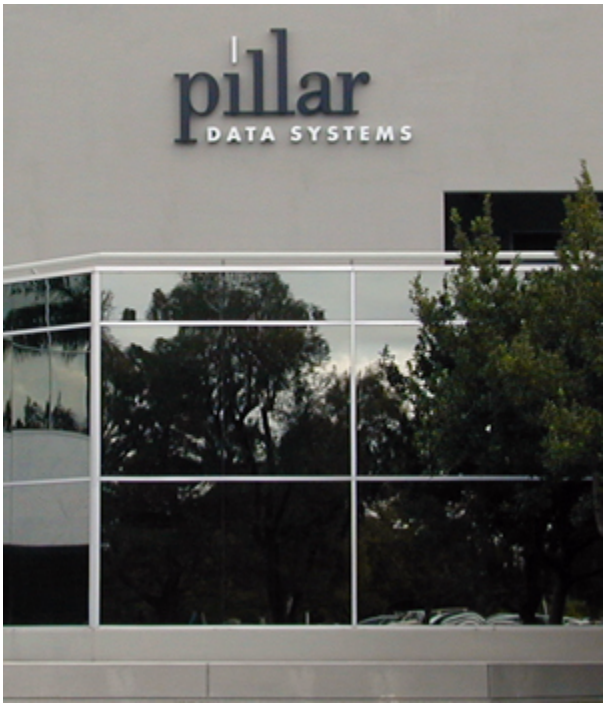
Return to the Bay Area

We returned to living in the Bay Area in the Fall of 2003. Anna had a job working for Mercury Interactive as a Technical Writer, and I began looking for a job in the storage software field. After some time spent interviewing, I got a job working at Pillar Data Systems as a senior software engineer in San Jose. I worked at Pillar from July 2004 to October 2005. I was strictly a coder with no leadership responsibilities.

Pillar Data Systems

In 2004, Pillar Data Systems was a startup working on its first product, the Axiom storage array. The Axiom was built, for the most part, from commodity hardware parts combined with custom software to make it into either a network attached storage server or a storage area network server.

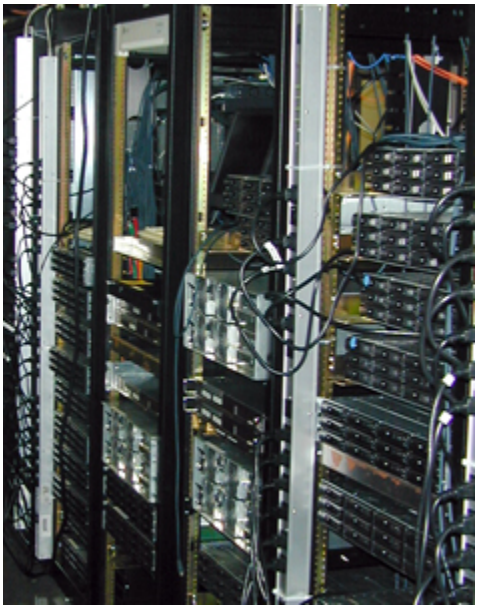
Even though it was billed as a startup, Pillar had only one large investor, Lawrence Investments, which belonged to Larry Ellison, the founder and CEO of Oracle. Lawrence Investments put more than $150 million dollars into Pillar before it released its first product in July of 2005. Pillar had many people working for it who had previously been at IBM SSD. This included their CEO Mike Workman, their 2nd in command Nance Holleran, and their hardware architect Mike Brewer. Pillar had two locations, San Jose and Longmont, Colorado. When I joined there were about 300 people at both sites.



What I did at Pillar

My job was to develop the multi-pathing software solution for the Pillar Axiom hardware attached to either a Windows or an AIX server. The software was written in the C programming language. This work was similar to what I had done at the end of my IBM career in San Jose. I viewed it as a challenge, because not too many software engineers could do this work on both server platforms. I was able to create the first versions of the software that Pillar shipped for these platforms. I was the third person who worked on the Windows version and wrote the AIX version by myself. What made the work especially challenging in the beginning was the fact that the Axiom hardware and software was very unstable. It would crash even when it was just standing idle doing nothing! Recovering from the crashes sometimes took hours of elapsed time. We spent a lot of time waiting for Axiom cold starts to take place.

The following picture shows some Pillar hardware in the lab I worked in. The boxes in the middle were called *"slammers"*, *"pilots"* and *"bricks."* IBM would call them control units and disk arrays.



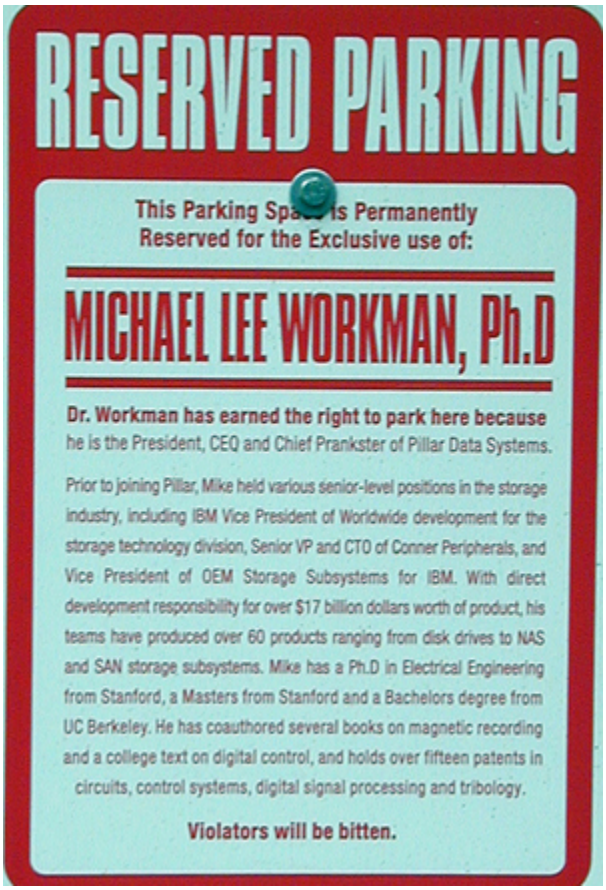
And here is my Pillar cube.

People I worked with

I had two managers at Pillar, John Klokkenga and later Dave Howard. My technical lead was Jeremy Farrell. Jeremy was from the UK. One developer I worked with quite a bit was Dave Powers.

This picture shows one of our execs' car in the Pillar parking lot. Notice the license plate and the sign on the pavement.

This was the parking place sign of the CEO, Mike Workman.



I left Pillar because I had seen the shipping of their first products and my code in July 2005 and because I wanted to do other things, like travel and work with Anna on other projects. This will be covered in the next section.

Pillar epilogue

In June of 2011, Larry Ellison's Oracle bought Larry Ellison's Pillar and an independent Pillar ceased to exist. As of this writing (March 2012), Oracle is not doing well in its efforts to sell hardware.

VR Communications, Inc.

VR Communications history

In 1993 Anna left IBM and started her own business, VR Communications, Inc., which has primarily worked on technical communications projects. In 1995 she converted the business to a corporation because it was hard to get consulting work as an independent contractor. The US Government had noticed that some companies were hiring people as contractors and keeping them on for a long time, making them into full-time employees, but without fringe benefits. If the companies hired another corporation, they did not have such a problem. Since 1993 VR Communications has done work for over 20 clients, including:

- Acta
- Avid
- CISCO
- Intuit
- Kyocera
- PayPal
- Pearson Foundation
- Pillar Data Systems
- San Jose State University
- Stanford University
- VMWare
- WITI (Women in Technology)

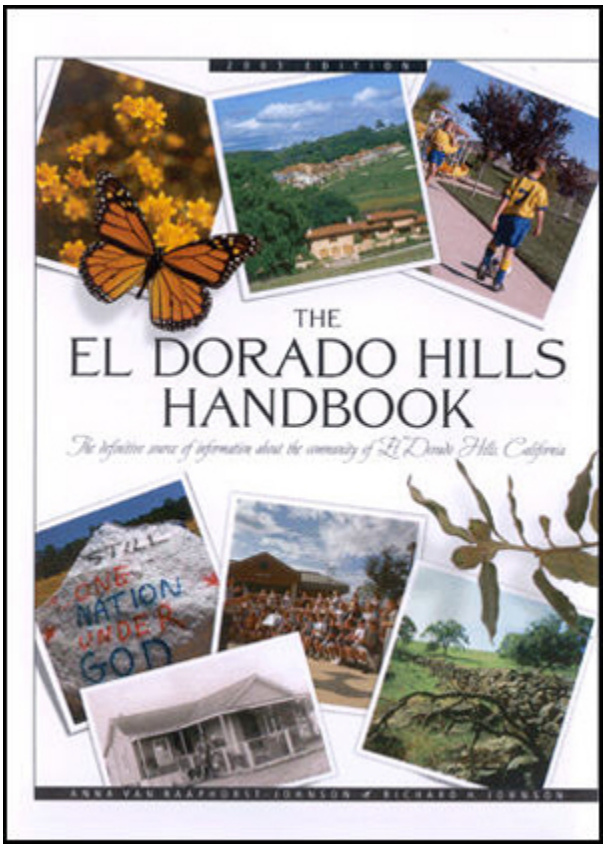
At times the business environment was such that companies did not want to hire contractors. In those times Anna took staff jobs instead of contracting. The preference for full-time versus contractor has gone back and forth many times since 1993. Some of the companies she worked for as a technical writing employee were:

- ISI (Integrated Systems)
- Siebel
- Skytide
- SLAC (Stanford Linear Accelerator Center)

The El Dorado Hills Handbook

In 2002 I left IBM and Anna and I moved to El Dorado Hills. We had hoped to do some high-tech consulting in the area, but that didn't pan out, so we decided to work together writing a book about El Dorado Hills that we called the *El Dorado Hills Handbook.* Even though there were thousands of people living there, there was no source of information about El Dorado Hills other than from the real estate developers. We made many field trips looking for information, going all the way from the Sierra to libraries in Sacramento.

The book was published in 2003 and was 312 pages long. We both did the research and photography, Anna did most of the writing, and I wrote a few of the topics myself. It was a fun project to work on, but we found that there weren't too many people in the community who wanted to buy a book about El Dorado Hills. We even ran into resistance from people who had been in the community for a long time and claimed they had planned to write just such a book themselves, but had not gotten around to it yet. You can still find copies of the book in the El Dorado County Library.



Here is Anna in the El Dorado Hills library with a copy of our book.

DITA projects

One of our first “second-career” activities (third for Anna, who taught high school for 10 years) was documentation for an open-source technology project: a user guide for DITA, an XML-based, end-to-end architecture for authoring, producing, publishing, and delivering information as discrete, typed topics.

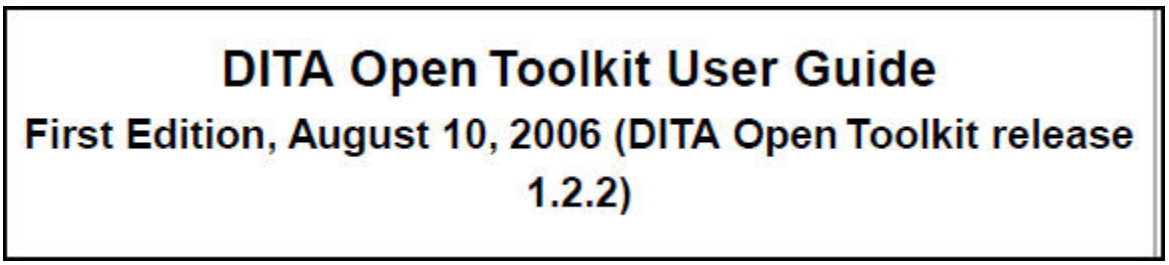
In March of 2006, Anna and I attended a DITA conference in Raleigh, North Carolina because we wanted to find out more about DITA, which was then a new standard for the format of technical documentation being worked on by OASIS, a standards organization.

DITA 2006
Thursday March 23, 2006 - Saturday March 25, 2006
McKimmon Conference Center
1101 Gorman Street
Raleigh-Durham, North Carolina 27606 Get Directions

An educational conference for DITA users of all knowledge levels. Come one, come all !

Most of the conference speakers were from IBM, since IBM created the first version of DITA for its own internal use. At the meeting we learned that there was a need for somebody to create user documentation

for the DITA Open Toolkit. The Toolkit was a reference implementation of DITA being done as a SourceForge project, mostly by IBM developers. Anna and I jumped at the chance to do this, since it would give us a chance to collaborate on a project that would use her writing skills and my software skills. We have been collaborating ever since! Together we managed to learn enough about DITA to publish the first edition of the *DITA Open Toolkit User Guide* in August 2006 using the DITA format.



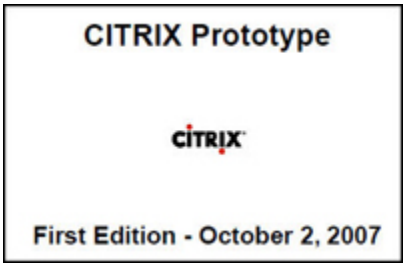
We approached this endeavor as a way to give back to our own technical community and also to gain skills in a promising new communication technology.

Once we published the DITA OT book, we had credentials that allowed us to land many DITA contracts between 2006 and 2010. We gave DITA writing team training seminars and did technical writing using DITA. We also gave presentations on DITA to meetings of the Society for Technical Communication and the Silicon Valley DITA Interest group (SVDIG).

This is a group picture taken at an SVDIG meeting.

Here are some of the DITA clients we worked with.

CITRIX - DITA training:



FedEx - DITA architecture and POC:

PayPal - team training:

Savi - team training:

SkyTide - DITA technical writing:

East Bay STC - presentation:

Pillar Data Systems - DITA technical writing:

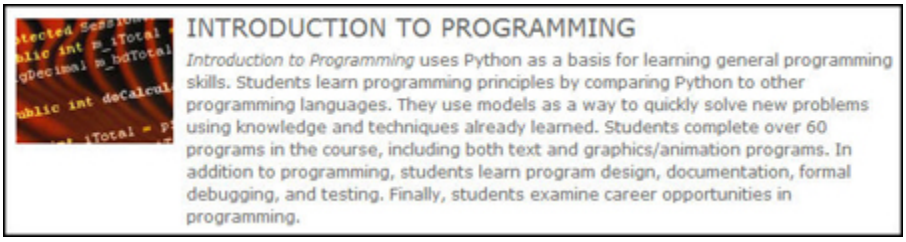
Pearson Foundation projects

In 2008 we spotted an interesting Internet job ad. The Pearson Foundation, a non-profit funded by Pearson Publishing (e.g. Penguin Books), was looking for a technical writer that also knew the Python programming language, to write a high school programming course for the National Academy Foundation (NAF). When we contacted them we learned that they had been looking for such a person for a long time and hadn't found anybody. So we proposed that Anna and I do the project as a team, Anna to focus on the writing and me to focus on the Python programming. The Pearson people were very reluctant to deal with two people instead of one, but finally decided to give us a try, since they had nobody else!

NAF fosters partnerships between the business and education communities to provide opportunities to underserved high school students. They have helped create a number of *"academies"* of industry-specific courses. Our class, Introduction to Programming, is part of the Academy of Information Technology.

After many months of hard and challenging work we produced the first version of the class. This included lesson plans, teacher resources, and student resources written with Microsoft Word, plus over 60 Python programs. Both Pearson and NAF ended up being very pleased with our work. They even got AT&T to sponsor the programming class.

In addition to revising the Python class several times, we have since helped revise and improve NAF classes on database design and web design.



Website technology

In 2010 Anna and I began to shift our focus from technical writing to websites. We noticed after the financial meltdown of 2008 that our opportunities for writing contracts began to decline. Many times during 2010 and 2011 we were contacted by companies that claimed they were interested in a project, but after much discussion, they silently went away and nothing came of it. Some of these *"clients"* just wanted free consulting. Prior to this, Anna had been maintaining several websites, so we began to consider whether we could include website building into our technical skills so we could do more than technical writing.

In May and June of 2011 we took a class from Sam Cohen (a Drupal expert) called *Mastering Drupal*. Drupal is an open source web content management platform. It is like an erector set for building a web site with lots of functionality.

It seemed to us that it might be possible to publish regular DITA content on a Drupal website, which would give you lots of capability to search it and display parts of it. You could also combine structured content

done by a technical writer with unstructured content written by a member of the community. This sounded like it would be fun and interesting to explore.

After we finished the class, we went on to build several model web sites to improve our skills and portfolio, and we even converted our personal site, News from Nan, to Drupal. With a little bit of Python script writing, I was able to publish structured data from our family trees to News from Nan.

As of this writing (April 2012), we have created several web sites as a Drupal portfolio. They are:

- www.ditainfo.info - a site about DITA containing both structured and unstructured information
- www.drupalinfo.info - a site about Drupal containing both structured and unstructured information
- www.edhinfo.info - a sample community site for El Dorado Hills
- www.newsfromnan.com - our personal family site. We are experimenting with publishing GEDCOM genealogy information about our family tree.
- www.rsf-earthspeak.org - cousin Andi's non-profit foundation site
- www.vrcommunications.com - our professional website
- www.wandbattorneys.com - law office site for daughter Jill and her law partner Kyle

Chapter 11. Looking back: Computers in the early 1960s

Introduction

A little over 60 years ago, I wrote my first computer program. I was a freshman at UCLA in 1960 and I was taking a beginner Fortran class offered by the UCLA Computer Club. The club arranged for our student programs to be run on the IBM 7090 that was installed at the Western Data Processing Center.

I have been thinking recently about how different the world of computers was then from what it is today and I wanted to describe how it was back then. The tools a programmer had in the early 1960s would be hard for someone to comprehend who is just starting in the field today. Read on to go back in time!

The period I am describing is from 1960 until about 1966. During that entire time the computer I was using was either an IBM 7090 (vacuum tube technology) or the IBM 7094, a compatible machine built with transistors. The computing environment did not change very much during this period, but that was because IBM was using almost all its resources to create System/360, which changed everthing!

Hardware and software

I wrote my programs for the 7090 and the 7094 using the Fortran programming language. The operating system was either IBM IBSYS or IBM FMS (Fortran Monitor System). The operating system only could do a few basic things:

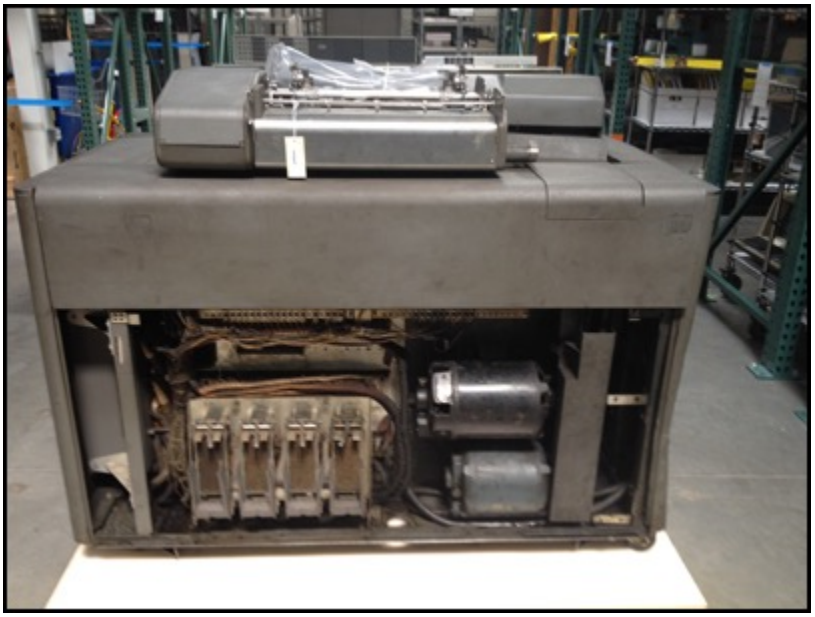
- Compile and run Fortran II or IV programs.
- Compile and run a Cobol program (I never learned or used Cobol).
- Compile and run a FAP or MAP assembler program.

The 7094 was designed for scientific applications. The instruction set included fixed and floating point arithmetic instructions that executed in a few microseconds each. Fast for the time. The machines had 32K 36-bit words of core memory. Each instruction took up one 36-bit word. Since each word held six 6-bit characters, that works out to about 196K of main memory in modern terms. The memory did not support parity checking for errors and there was no hardware protection of the operating system, so each program that ran had complete control of the entire machine. In modern terms, the machine always ran in "supervisor" state. Programs were only run one at a time. A running program could easily crash the machine. When that happened the operator could load a small program from cards to generate a memory dump written to tape.

The machines cost about $3,000,000 in 1960 and took up considerable floor space in an air conditioned room.

Here is a picture of the console of a 7094 now residing in the Computer History Museum warehouse. An entire machine filled up a large room (usually with a glass front wall) and included the 7094 itself, the console, tape drives, a line printer, and a card reader/punch. Usually nearby was an IBM 1401 computer as well.

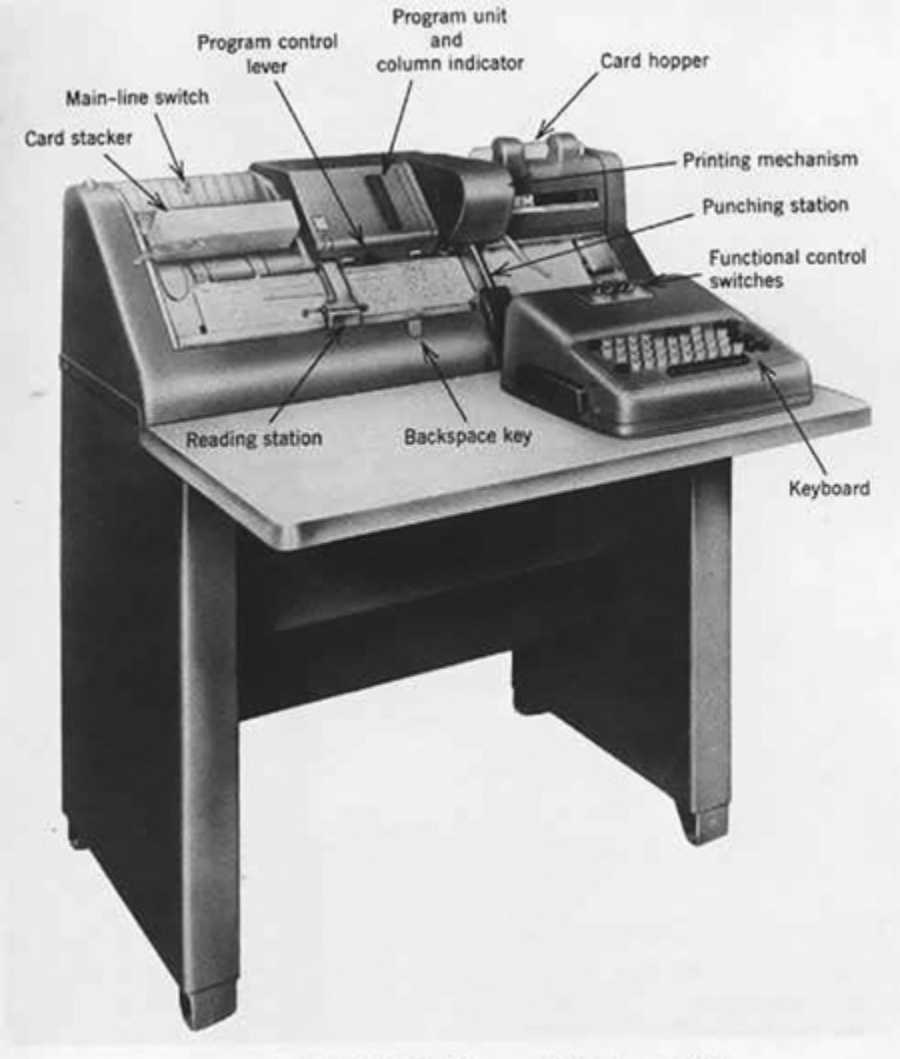
This next picture shows the IBM 716 line printer with the covers off (also in the Computer History Museum). It would have been attached to a 7094. This device alone weighed over a ton and as you can see, was an electro-mechanical marvel. It could print 150 lines per minute and was first announced in 1952.



Punched cards, tapes, listings and the 1401

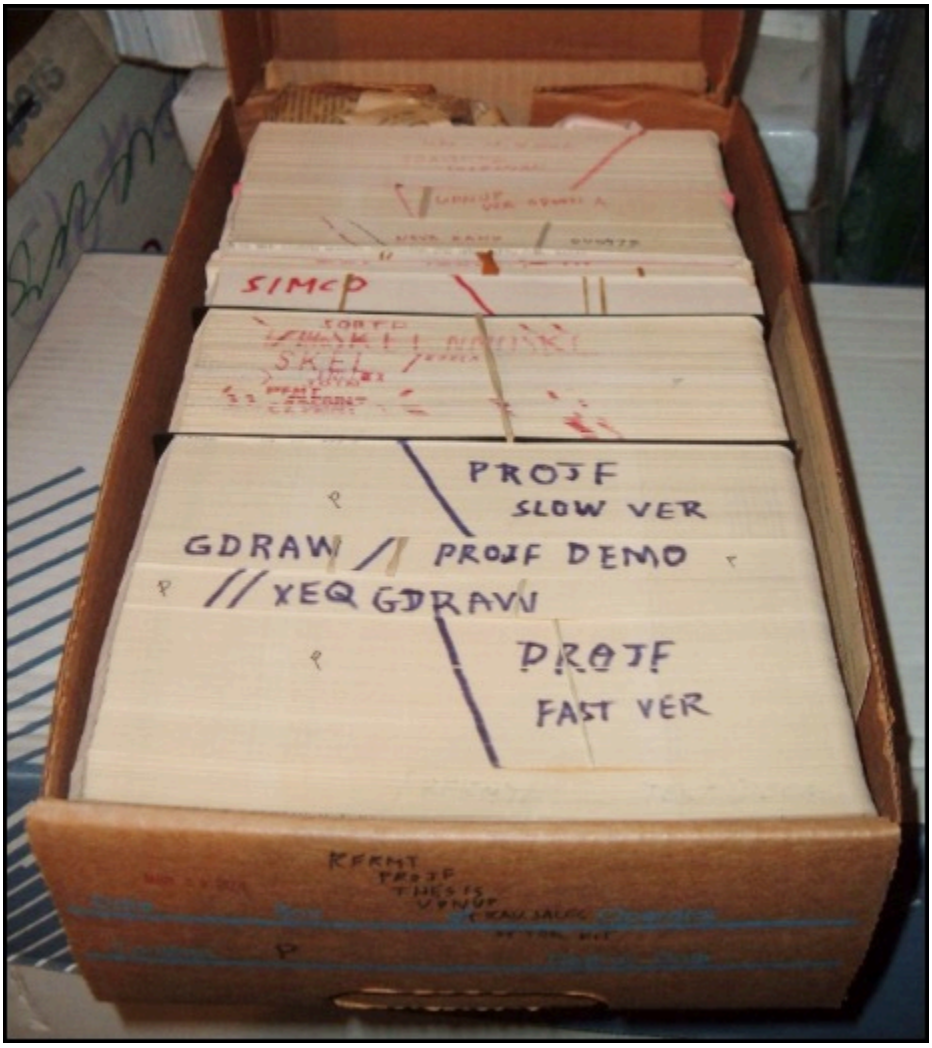
All programs and data were stored either on punched cards or on reels of magnetic tape. A punched card had 80 columns for data and a reel of magnetic tape could hold up to 5MB of data on a 2400' reel. All data was accessed sequentially. There was no random access to data unless it was in memory.

Below are pictures of an 026 key punch and a 729 tape drive of the era.


Program unit
and
column indicator
Program control
lever
Card hopper
Main-line switch
Card stacker
Printing mechanism
Punching station
Functional control
switches
Reading station
Backspace key
Keyboard

All the programs I wrote were entered on punched cards using an IBM 026 key punch machine. (You can still see an 026 in operation at the Computer History Museum in their 1401 exhibit.) As a UCLA student, I did all the keypunching myself. At my first job with IBM Service Bureau, there were people who did the keypunching for us after we wrote our programs or data on keypunch forms. After punching the cards, we could print them by using an IBM 407 card reader/printer that was near the key punch machines. The listing would be examined carefully to check for typos before submitting the cards as a batch job. We only had a few chances per day to run our programs!

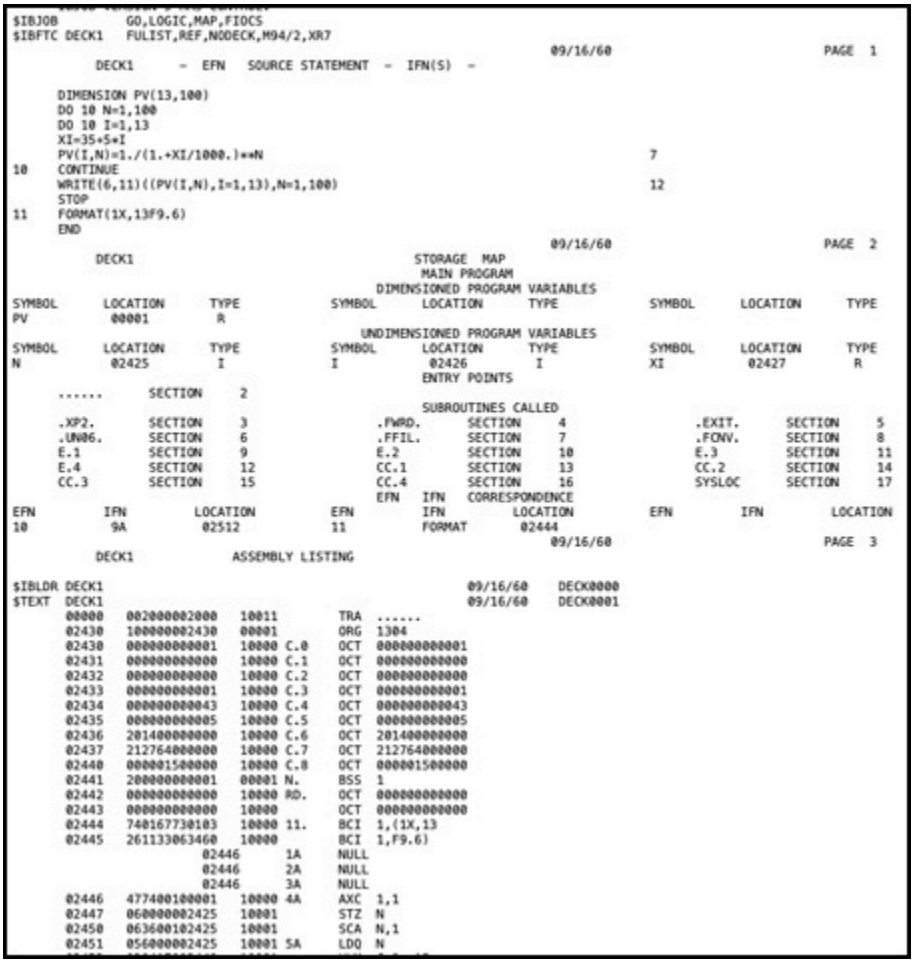
This photo below that I found on a Columbia University web site shows how we stored program decks in punched card boxes. At one time, production of these cards was using up a lot of cardboard worldwide! The diagonal lines written in felt pen on top of the decks were used if the program deck got dropped and needed to be put back in sequence.



The 7094 tape drives were the IBM 729 II. They were cleverly designed with vacuum columns to hold the tape media so tape movement could be started and stopped quickly.

Transfering the cards to tape was done on a IBM 1401 computer in the same room with the 7094. The 1401 was also used to output punched cards and printed listings produced on tape by batch jobs run on the 7094. Normally the card punch and printer attached to the 7094 were not used for output, since they were very slow and 7094 time was very scarce and expensive.

The 7094 used the 6-bit BCD (Binary Coded Decimal) character set. There were only 64 possible characters, and they did not include any lower case or graphics. This is why when you look at old printed computer listings, it is all upper case. Below is an excerpt from a 7094 listing.

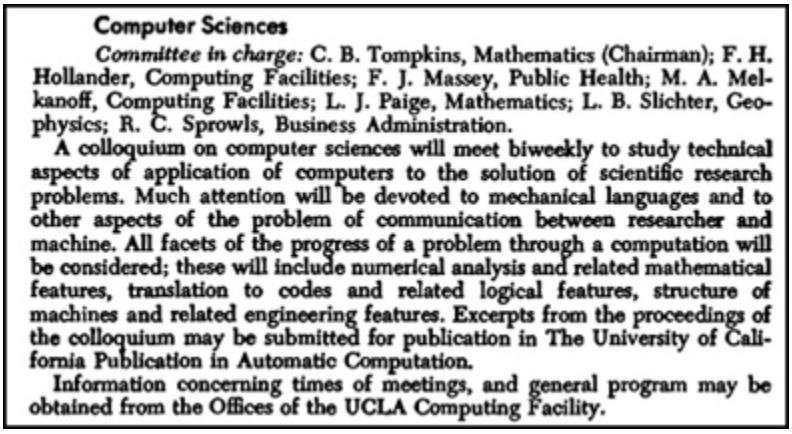


```
$IBJOB        GO,LOGIC,MAP,FIOCS
$IBFTC DECK1  FULIST,REF,NODECK,M94/2,XR7
                                                                09/16/60                        PAGE  1
          DECK1      -  EFN   SOURCE STATEMENT  -  IFN(S)  -

     DIMENSION PV(13,100)
     DO 10 N=1,100
     DO 10 I=1,13
     XI=35+5*I
     PV(I,N)=1./(1.+XI/1000.)**N                                          7
10   CONTINUE
     WRITE(6,11)((PV(I,N),I=1,13),N=1,100)                                12
     STOP
11   FORMAT(1X,13F9.6)
     END
                                                                09/16/60                        PAGE  2
          DECK1                                  STORAGE  MAP
                                                  MAIN PROGRAM
                                               DIMENSIONED PROGRAM VARIABLES
SYMBOL      LOCATION      TYPE          SYMBOL      LOCATION      TYPE          SYMBOL      LOCATION      TYPE
PV           00001         R

                                             UNDIMENSIONED PROGRAM VARIABLES
SYMBOL      LOCATION      TYPE          SYMBOL      LOCATION      TYPE          SYMBOL      LOCATION      TYPE
N            02425         I            I            02426         I            XI           02427         R
                                                    ENTRY POINTS
     ......       SECTION     2
                                                    SUBROUTINES CALLED
     .XP2.        SECTION     3            .FWRD.       SECTION     4                .EXIT.       SECTION     5
     .UN06.       SECTION     6            .FFIL.       SECTION     7                .FONV.       SECTION     8
     E.1          SECTION     9            E.2          SECTION     10               E.3          SECTION     11
     E.4          SECTION     12           CC.1         SECTION     13               CC.2         SECTION     14
     CC.3         SECTION     15           CC.4         SECTION     16               SYSLOC       SECTION     17
                                           EFN   IFN   CORRESPONDENCE
EFN          IFN          LOCATION        EFN          IFN          LOCATION        EFN          IFN          LOCATION
10            9A           02512          11           FORMAT        02444
                                                                09/16/60                        PAGE  3
          DECK1            ASSEMBLY LISTING

$IBLDR DECK1                                                    09/16/60    DECK0000
$TEXT  DECK1                                                    09/16/60    DECK0001
       00000   002000002000   10011        TRA  ......
       02430   100000002430   00001        ORG  1304
       02430   000000000001   10000 C.0    OCT  000000000001
       02431   000000000000   10000 C.1    OCT  000000000000
       02432   000000000000   10000 C.2    OCT  000000000000
       02433   000000000001   10000 C.3    OCT  000000000001
       02434   000000000043   10000 C.4    OCT  000000000043
       02435   000000000005   10000 C.5    OCT  000000000005
       02436   201400000000   10000 C.6    OCT  201400000000
       02437   212764000000   10000 C.7    OCT  212764000000
       02440   000001500000   10000 C.8    OCT  000001500000
       02441   200000000001   00001 N.     BSS  1
       02442   000000000000   10000 RD.    OCT  000000000000
       02443   000000000000   10000        OCT  000000000000
       02444   740167730103   10000 11.    BCI  1,(1X,13
       02445   261133063460   10000        BCI  1,F9.6)
                         02446      1A     NULL
                         02446      2A     NULL
                         02446      3A     NULL
       02446   477400100001   10000 4A     AXC  1,1
       02447   060000002425   10001        STZ  N
       02450   063600102425   10001        SCA  N,1
       02451   056000002425   10001 5A     LDQ  N
```

Notice that there is only one type face in the listing, and no fonts, boldface, underlining, etc.

Education and reading resources?

When I entered UCLA there was no Computer Science major, and only a few classes offered that even mentioned computers. By 1965, when I left, there was a Computer Sciences colloquim meeting bi-weekly, but still no CS major. The image below is taken from the UCLA General Catalog of 1965 and it describes the colloquium.



Computer Sciences

Committee in charge: C. B. Tompkins, Mathematics (Chairman); F. H. Hollander, Computing Facilities; F. J. Massey, Public Health; M. A. Melkanoff, Computing Facilities; L. J. Paige, Mathematics; L. B. Slichter, Geophysics; R. C. Sprowls, Business Administration.

A colloquium on computer sciences will meet biweekly to study technical aspects of application of computers to the solution of scientific research problems. Much attention will be devoted to mechanical languages and to other aspects of the problem of communication between researcher and machine. All facets of the progress of a problem through a computation will be considered; these will include numerical analysis and related mathematical features, translation to codes and related logical features, structure of machines and related engineering features. Excerpts from the proceedings of the colloquium may be submitted for publication in The University of California Publication in Automatic Computation.

Information concerning times of meetings, and general program may be obtained from the Offices of the UCLA Computing Facility.

Interestingly enough, C.B. Tompkins, listed above, was my graduate advisor, but I don't recall he even mentioned the colloquium to me.

In the early 1960s many colleges and universities believed that classes teaching about computers were more suitable for trade schools than for institutions of higher education. My first computer education came from attending classes given by the UCLA Computer Club. The club was organized by some of the students.

Also in this era, there were very few books available on computers or programming languages. There was a series of books written by Daniel D. McCracken on various programming languages. I had one called "A Guide to Fortran II". IBM did publish a set of manuals on the 7094 and the various programming languages that ran on it.

What we didn't have

Here is a list of things I have come up with of software, hardware and tools that are common now that did not exist in my computer world of 1960-1966.

- There were no file systems or even file names. I/O in a 7094 Fortran program was directed to a unit number, which usually translated to a reel of tape on a physical tape drive.
- There was no random access I/O. All data was read and written sequentually.
- There were no databases.
- There was no virtual memory. A program had to fit entirely into real memory. There was the ability for a program to "chain" load another program from tape to replace it.
- There were no multi-processor machines.
- There was no supervisor/user state distinction. Each program had complete access to the entire machine and could damage the operating system if it had the right bugs. Array bounds overruns were a common way to crash the machine.

- There were no code manaagement software systems.
- There were no text editors.
- There were no terminals and no time-sharing.
- Our program printed output did not contain mixed case or graphics.
- And lastly, there was no network, email, or file transfer and of course no internet!

In the ten year after 1966, all the items listed above came into being.

Index

D

E

F

G

H

I

R

S

T

U

V

W

X

Y

Z