

I Guess That's Me (A Reflection)

Lee Frank

Computing

Copyright © Lee Frank 1998-2003
All Rights Reserved

[Next Chapter](#)
[Previous Chapter](#)
[Table of Contents](#)
[Memoir Home Page](#)

Computing

By the end of the year (1963) my first real job ended. I was now head of indoor sales, and like Wayne before me, I wanted a few more pennies, commensurate with my increased duties and responsibilities. My military obligation over, my college plans discarded, my first real love ended, my life was ready for a new direction. It didn't come right away. I went back to my studies, halfheartedly looking for another job. Fortunately, I didn't find one. Hence I was unencumbered when my cousin Ron asked me what I was doing that summer. I said nothing and he asked if I might be interested in learning how to program computers. There was, he said, this new computer language called FORTRAN and it was, he said, easy for someone with my engineering background. I was only mildly curious but all it required was reading a book on the subject. I was reading lots of books on lots of subjects. Why not this one?

I was hooked well before I finished the book (for those who remember, this was the McCracken primer). FORTRAN stands for Formula Translation. A simple FORTRAN statement was $A = B + C$, meaning simply what it says: Add B to C and place the sum in A. But more than straightforward mathematics, FORTRAN was highly logical. IF this THEN that, ELSE some other. With

one additional step, these statements controlled the computer. (That step, another program called a compiler, turned the FORTRAN statements into small, discrete, specific machine instructions.) Immediately I could see endless possibilities. I finished the book in short order and told him I was very interested. He asked a few questions to see if I really understood. Then he gave me an overview of the practicalities: How this instruction might be faster than another, how compilers on different machines varied, and so on.

Then he took me into his office. This was IBM's tiny Math and Apps division (Mathematics and Applications) in the middle of Manhattan. If you count a Bachelors, a Masters, and a PhD as three separate degrees, than each office there averaged 2.75 degrees—with a single occupant. Attached to this little group were people like the legendary Ken Iverson, creator of the APL programming language. Others were names I came to know later though the CACM (Communications of the Association of Computing Machinery), the professional programming magazine. Here I was, something like an intern, and thoroughly wet behind my computing ears.

Ron showed me the basics of submitting jobs (we used a large IBM computer in another building). This was the heyday of punched cards and each job required a set of written instructions (primarily checking certain boxes on a cover sheet) for the operator. It was also the era of the one day turnaround. For those of you—and myself here typing—familiar with desktop computers and instant responses, this may be hard to imagine. We'd prepare a job; a messenger would bring it to the computer; another messenger would return the next day with the results. If you'd made a mistake, either in the job or its preparation, you wouldn't know until the next day. You knew by looking at the printout. If things went wrong, the machine printed every bit of the information it had (everything that was going on inside it at the time) to aid you in finding the

problem. That occasionally meant receiving a stack of paper a foot high.

I screwed up once. Ron looked at the first page and pushed the foot-high stack of paper into the wastebasket. (Today, I have a plastic thirty-gallon garbage can as my office wastebasket. So much for the paperless society.) He then reached into the punched card deck and pulled out the offending card. I'd put it in backwards. Instant lesson: In computing, you can't be too careful. Later lesson: I discovered by putting a "C" in the *last* column on a card I could keep it in the deck simply by reversing it. A "C" in the *first* column made it a comment card, ignored by the compiler. This way I was able to retain my previous FORTRAN statements in the deck as comments until I was sure I didn't need them.

Perhaps you recognize another lesson here. The one from Architecture class about never throwing anything away. Some people accuse me of being a pack rat. Perhaps, but only in this way: I won't throw it away until I'm sure I don't need it anymore. Since I'm not, currently, efficient at reviewing this saved material, it tends to hang around a bit. But when I do review, if it's not needed, out it goes. That thirty-gallon wastebasket fills up rather quickly. (If I didn't save anything, I wouldn't have the photos for this book.)

It's hard to explain the change computing brought about in me. Here was a field still new, work I enjoyed and was good at, and an apparently unlimited future. I quickly learned from this Math and Apps group that computing skills were separate from the knowledge of these learned people. They might be the experts in understanding the problems they were trying to solve, but a separate and equal expertise was needed to solve problems on the computer. The key to connecting these

disparate skills was communication. While the computer programmer could never understand a problem to the same degree as an expert in some discipline, he or she had to know, or learn, enough about that discipline to solve the problem. I'm not saying many of these people were not also computer experts. They were. But it was an ability distinct from their primary calling. Some, like my cousin Ron, wandered from their principal interest, in his case mathematics, to become uncommon computer experts.

More importantly, I, regardless of my education and background, could become one of these experts. Yes, there were many technical things to be learned, but at this time there were few schools offering degree programs. And the field was growing so fast employers looked for experience before education. Four years later, I dropped out of school (for the second time) because a degree offered less advancement than choosing the right job. But I get ahead of myself. Here, in 1964, I was entering a brand-new field anyone could foresee as having unlimited potential. In the succeeding years, its rapid expansion has often been called explosive. Here I was, unexpectedly, at the threshold of not just my future, but *the* future.

Side bar on luck. At this point, you may perceive this emerging theme of luck. Clearly, my entry into computing was as lucky as one could get. Without any particular prerequisite, here I am learning at one of IBM's most exclusive enclaves. And my next job, as you'll see, is even better. How I acquire the one after that, will seem even luckier. What's the deal with this luck? Don't know, but let's review. I get into the college of my choice when I had no options and had not worked hard to qualify. I not only survive the Army, I do it in style. Could I have been luckier? Perhaps Paris? Not really. By returning home I was able

Computer Me

to keep seeing the girl I loved. And was not having someone like her love me the very acme and zenith of luck?

These are examples of good luck. They exist in a universe neatly balanced with bad luck. For me, the bad includes my arthritis. If you want an example of really bad luck (not mine), check this out. In less than three years, four different television series lost five key supporting actors. Three of these died within three months of each other, two within the same week. The details: Michael Conrad of Hill Street Blues (22 Nov. 1983); Nick Colasanto from Cheers (12 Feb. 1985); Dolph Sweet of Gimme A Break! (8 May 1985); and from Night Court, Selma Diamond (13 May 1985) and Florence Halop (15 July 1986). And four of these people played uniformed police officers. If you want more coincidence, look up their causes of death. Finally, these series were all on NBC!

[Next Chapter](#)
[Previous Chapter](#)
[Table of Contents](#)
[Memoir Home Page](#)