Programming with Punched Cards

dalefisk@gmail.com

It must have been about 1973. Life at IBM was good, and I was busy doing whatever it is that engineers did then. Suddenly, in the life of our project, something came up that called for a computer program that did not exist, and I was asked to create it. My boss knew I'd never written a program before; not unusual since in those days there were very few engineers who knew how to program.

I'm sure he thought about asking a real programmer to do the job. Maybe he couldn't find anyone available. Or maybe he decided the program would be so technically involved that it would be easier for an engineer to learn how to program than it would be for a programmer to learn how to be an engineer. In hindsight, I think he made the right decision, but at the time he was taking a huge gamble. Could I learn enough, quickly enough, to be the right choice?

He told me how much confidence he had in me and that he knew I would do an outstanding job and that it would be a wonderful learning experience for me. He patted me on the back (probably with his fingers crossed behind his own back) and sent me on my way.

The first thing I needed to do was to learn how real programmers programmed. I knew that down the hall was a big room that had two brand new IBM 360 computers, and I knew that the programmers kept walking in and out of that room carrying big, flat boxes of punched cards. The boxes they carried actually looked more like a tray – picture a box about three inches high, eight inches wide, and maybe fifteen inches long.

Walking past programmer's offices I could see that they had stacks and stacks of punched cards and boxes and boxes of them in their offices. They had nice, neat boxes stacked in corners and ragged-looking ones poked under tables. Some boxes had pink cards, others held blue cards, and more had those boring cream-colored cards. Some of the stacks of cards, as well as the boxes, had seemingly random collection of different colored cards. All had names, usually cryptic, written on the sides of the cards with a felt-tipped pen. I had seen them flipping through the cards, looking at them, replacing them one by one. This was programming!

Figure 1 Punched card program decks

I learned that each punched card described one instruction in the program and that each box could hold about two thousand cards. The size of a box, then, became a built-in limit to the size of a program. More than one box was awkward for a guy (or gal – we had lots of women programmers) to carry around. One full box was pretty heavy; more than one became a load.

There was another reason to keep this number of cards as a limit. The punched card hopper on these brand-new computers could hold just about that amount at one time. Any program requiring a more than one box required that the computer be spoon-fed the boxes in the right order.

My program shouldn't be too big, so I wouldn't have to worry about managing more than one box of cards. I hoped.

Writing a program began with a paper tablet of coding forms. Each page of the tablet had about fifty lines on it, and each line on the form would eventually be converted into a punched card and stowed away in a box with a bunch of other cards

So that's how I began. I went down to the big metal stationery cabinet that Bob, my programmer buddy, had showed me and picked up a pile of coding form tablets to bring back to my desk. I was a programmer! With my **How to Write a Program** book open at my elbow, I started writing my program. I wrote and erased and wrote some more, and finally I had a stack of dog-eared sheets that I thought should make at least the beginnings of a program.

I knew what to do now. I took my coding sheets down to our computer lab. Well, actually I went around to the back door because that's where our keypunch operator worked. She had a room just off the computer lab where she and her keypunch machine resided. I greeted Maria, who I already knew as a wonderful lady with a ready smile, a quick wit and fingers that just flew over the keyboard. She showed me the 'in' basket and told me to come back tomorrow to pick up my deck of cards.

Deck? Like a deck of playing cards? I had expected at least a partially filled box! No way could everything I had done be contained in a 'deck!' But, sure enough, when I went back the next day Maria's 'out' basket held my dog-eared coding sheets wrapped around a half-inch high stack of punched cards, all held together by a rubber band. Chagrined at the meager size of my card stack, I looked around to see if anyone was watching before picking them up.

I took them back with me and carefully set the brand new deck of cards down in the middle of my desk. They sat there quietly, challenging me to do something with them.

What to do next? Maybe I could postpone the next step by playing with these new cards a bit. I picked up one of the cards carefully, not wanting to 'fold, spindle or mutilate' it. Maybe I should make sure that Maria had punched them correctly? I could compare the cards with what I had originally written on my coding sheets. That sounded good.

And then, another rude shock. The card I was looking at had little square holes punched in it as I expected but nothing that told me what the little square holes meant. I knew there was a code to translate from the punches to each specific letter in the alphabet, but manually decoding those punches would take forever! This seemed serious! What if I needed to change something? I flipped through the rest of the cards and sure enough none of the cards had any writing on them at all. I took the deck of cards back to Maria and asked her how to find out what had been punched into each card.

"Oh," she said, "You didn't ask me to turn on the keypunch printer. If you had, the keypunch would have typed out on the top of each card what the card was saying." Ever helpful, though, she took the deck of cards and put them into another machine the size of a small desk. "This is a punched card interpreter." She said. "It will put that typing you want on the top of each card." And, a couple minutes later she returned the deck of cards to me.

Sure enough, each card now had typed on the top of the card a line of text that agreed with one line from the coding sheet I had given her.

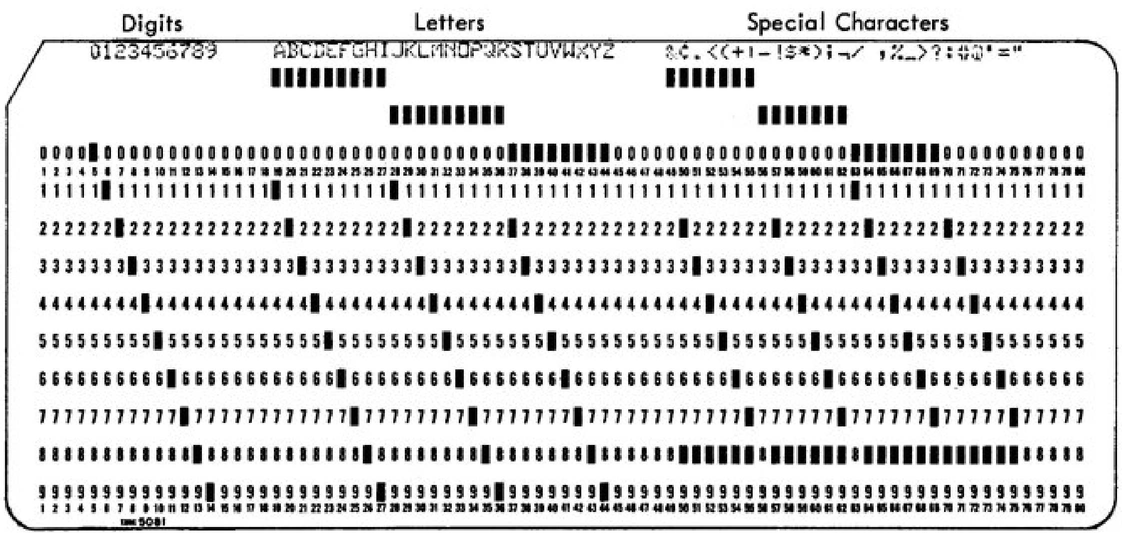

Figure 2 Punched card character coding

Now what? I knew the next step was to feed the deck of cards into our IBM 360 computer. Clearly I needed a little guidance, so I went back to Bob and asked what I should do next with my program.

"First," he said, pointing to the small stack of punched cards I had set on his desk, "That deck of cards isn't yet a program. That deck of punched cards is what we call a source deck. Your source deck is a description, very detailed and very technical, but still just a description of what you want your program to do."

“Your source deck,” he said, “has to be read by the computer, and a program called a compiler will look at each card. When it is done looking at all of them the computer will punch out another deck of cards. That second deck of cards will be your program.”

Then I understood. Writing a program is a two-step process, where first a program called a compiler, already written by somebody else, would read my ‘source deck’ and create another set of punched cards, the ‘program.’ I would then ask the computer to read my new deck of cards, my program, and the computer would do whatever it was that I was asking it to do!

Following Bob’s instructions I took my deck of cards, the source deck, back to the computer lab, this time going to the front door. Actually, by the door was an open window into the lab with a big counter under it, with an ’in’ section and an ‘out’ section.

Only a select few programmers were allowed in the computer lab. The rest of us were barred from that temple to technology, so I filled out a request form and laid it, along with my source deck, on the ‘in’ section. I saw several other similar stacks of punched cards there, and a couple big cardboard boxes as well. They were ahead of my request, so I knew I would have to wait for a while. How long? I didn’t know.

I knew what would happen next. The computer operator, a young guy that spent his entire day inside that mysterious room, would at some point gather up my program deck and walk over to the computer’s punched card reader, a machine the size of a small desk that had a big electrical cable running from it over to the actual computer. I knew which was which because the through the open window I could see that the computer had a bunch of flashing orange and green lights on the front, as well as a bunch of dials and toggle switches that I had once seen the computer operator twist and flip.

Figure 3. Computer console is on the far left. The punched card reader is against the back wall on the right. The cables connecting the various devices run beneath the floor.

Sometimes he would sit down at a special typewriter – a Selectric typewriter that he used to control what the computer did. Remember the Selectric typewriter with its golf-ball type element that danced and spun its way across the page? This particular one was connected directly to the computer and only the computer operator was allowed to use it. A few keystrokes on this typewriter would stop a program. A few more and a new program would take its place! That was one special typewriter.

I stood in front of the 'in' counter for a bit, watching through the window as the orange and green lights flashed on the computer console and the computer operator moved things around inside the computer lab. He picked up a big box of paper destined for the printer and disappeared behind the computer. When he didn't reappear I reluctantly I went back to my office. Half an hour later I checked again, and my request form, and source deck, still lay in the 'in' section of the counter. So did all the other decks of punched cards I had seen earlier, along with a few that had been added since I was there last. Disappointed, I went home.

The next morning I eagerly went down to the computer lab – the 'in' section of the counter was empty! Not the 'out' section, though. Sorting through the various program results there, I found my source deck and a stack of computer printout paper with my name on it. I picked up my source deck and the stack of paper (actually, it was only about an eighth of an inch thick) and took them back to my desk to see what I had. It wouldn't be cool to stand at the counter and read it.

The first thing I did was to admire the first page of the computer printout where my name was blazed across it in three-inch letters. No need for reading glasses here! Lifting the first page of the stack of fanfold paper, I studied the first of several pages telling me in great detail how the computer had first understood what program I wanted to run -- the compiler -- and what to do with the result -- print out a report and create a punched card deck containing the program.

Then it dawned on me. The computer had not created that program deck that I told the compiler program to produce! Dumb computers! I had this small stack of paper but no punched cards. Clearly the computer had messed up somehow. I'd have to fix that.

I turned a couple more pages of the computer printout and finally found the section where the compiler started to process the source deck. And, the compiler immediately did the equivalent of a computer barf! An error message described in infinite detail some problem. The computer obviously had hiccupped and done the wrong thing! The only thing to do was to rerun the program – the compiler – and give the computer a chance to do it right this time.

I took my source deck back to the computer lab and called the computer operator over, telling him that his computer had messed up and that I wanted him to redo the request I had originally put in. He said: "Are you sure?" I was. "Just as soon as the program running now finishes," he said.   He took a stack of punched cards out of the card reader, set them aside and put mine in their place. I impatiently waited for the present program to finish so the computer would redo my source deck. I remember wondering a trifle guiltily whose cards the operator took out of the card reader in order to redo my request.

It didn't take long before the other program finished and mine started. Then, almost immediately, I got another stack of paper back. Again no program deck! In fact, the one-eighth inch stack of computer printout showed the exact same error message I had seen earlier.

Could it be that it wasn't the computer? Something that I had done? I trudged back to my desk and glumly pulled out my **How to Write a Program** book again. And, after a while I found that one of the cards in my source deck was missing a needed comma. I remember grumbling that if the computer could figure out that a comma was needed, and exactly which comma was missing, why didn't it fix it for me? Dumb computers!

So now I dug out my pad of coding forms and wrote one line on it, this time including the missing comma. I took it back to Maria and asked her to make me a new card. I remember that she looked at me kinda funny, but she really was a nice lady and interrupted what she was doing to and made me a new card. One card! I took it back to my desk and pulled out my source deck. I found the old card with the missing comma, threw it in the wastebasket and inserted the corrected punched card in its place.

Back to the 'in' counter I went, waited a while and retrieved the computer output. Again there was an error message, but a different one this time. I studied my source code to see what went wrong and then went back to Maria with my one-line coding form. I didn't blame the computer this time. Maria looked at my one-line coding form and said: "Let me show you how to use the spare keypunch machine. You can usually make these one-card changes quicker yourself." In addition to becoming a programmer I was learning how to use a keypunch machine**.**

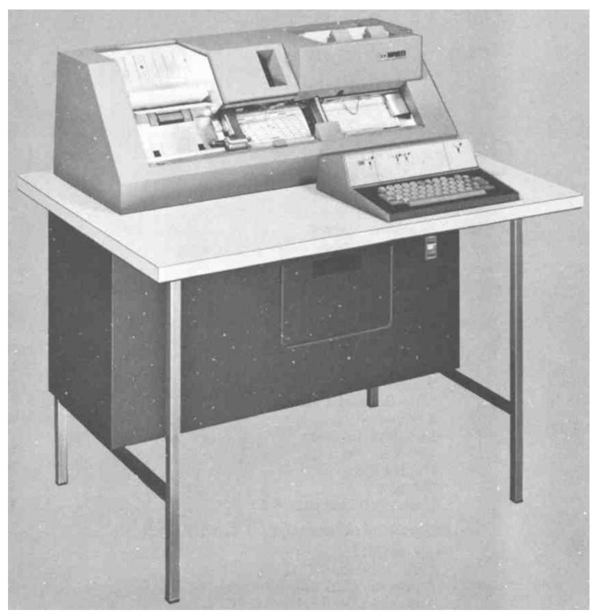

Figure 4. IBM 029 Keypunch Machine

Now things went a little faster. I could make my own corrections without waiting for Maria to finish what she was doing. Instead of two or three corrections a day I was now able to run the compiler program four or five times! In fact, a few days later the 'output' counter had, in addition to a half-inch stack of computer printout paper, TWO decks of cards! It held my source deck, now getting a little grimy around the edges, and a brand-new set of cards that was clearly my program deck!

I had reached a milestone in my journey toward becoming a programmer. I now had a program that I could run. I picked up both decks of punched cards, along with the computer printout and almost ran back to my desk. This time I decided to look closely at the program deck. By now I knew just about everything there was to know about the source deck.

The program deck was quite different from the source deck. To start with, it was a much smaller stack of cards. There were no letters typed on the top of the cards, and there were lots of punches in each of the cards. Lots of punches! In fact, some of the cards looked almost like lace. Clearly in order to verify that the compiler had done its job right I would need to learn how to read these cards as well as I could now read the source deck cards. I took the program deck to Bob and asked him how to decipher all those punches.

“Don’t worry about it,” he said. “This deck of cards, your object deck, is punched with a binary coding system that only computers understand. You could figure it out if you really had to, but you don’t.”

I couldn’t put it off any longer. I had to run my program. So I set my source deck of punched cards up on a shelf in my office and walked down to the computer lab. I filled out another request form and left it, along with my brand-new program deck, on the ‘in’ counter.

The next morning there was another stack of computer printout paper, and my program deck, on the ‘out’ counter. I remember a feeling of awe and pride – my very own program! I carried both back to my desk and put the program deck up on the shelf next to my source deck. The printout paper went smack in the middle of my desk, and again my name, in big block letters, proclaimed my ownership.

I flipped through the first several pages of what I had come to think of as computer boilerplate before I got to where my program started to control the computer. I had asked the computer to print, in this listing, what it was doing, and sure enough, it had. My program was working! I stared at it with wonder.

But was the program working right? Probably not. By now I now knew that programmers spent a lot of time ‘debugging’ programs. So I fully expected that my program would have errors, or bugs, in it, and, it did. Lots of bugs. Some were big, some were small, but all bugs that I needed to fix. So, how did I fix all those bugs?

I started by studying the compiler program printout, a list of all the instructions in the program, to see what was causing the bug. Then I figured out what the program should have done, marking up the source code printout with corrections that needed to be made. I copied these corrections onto a coding form, took that coding sheet down to the keypunch machine, sat down at the spare keypunch machine and created new punched cards to replace incorrect ones in my original source deck.

Finding the incorrect punched card and replacing it with a new corrected card became more and more difficult as my program grew. A few times I replaced the wrong card, introducing a brand new bug into my program without fixing the old one. I learned to be very, very careful when I updated my source deck.

I would take my updated source deck back to the 'in 'counter and wait for the output to appear on the 'out' counter. Usually, the compiler output now included a brand-new program deck, so I would move six feet to the left and fill out a new request form with my new program deck before I went back to my desk.

One day I was asking Bob something about my program and he said: "Now that you are debugging your program, why don't you combine your two computer runs? Run the compiler and then follow that by running your program all with one request form? "

He showed me how, and the number of times I could run my program each day doubled!

I was becoming less aware of the debug process and more focused on what my program was supposed to do. The sophistication of the program increased, and my source deck became several inches high. Or long – I had adopted the habit of laying the cards, wrapped with a rubber band, on their side. The pile seemed more stable that way.

Over the next several weeks the amount of wear on the punched cards accumulated, and one day the computer operator wouldn't take my source deck, telling me "The card reader on the computer can't read it any more. You will have to get a new set of punched cards."

Now what? I only had one deck of source cards. Backups? Who needed a backup when your program source code physically sat on your desk? Nobody had a backup.

It had been a while since I had visited with Maria, but clearly I needed her expertise. And she knew exactly what to do. "The computer card reader is sensitive and will only accept punched cards in very good condition. However, we have another machine, a punched card duplicator, designed to do just what you need to do. Give me your source deck."

She took my gray, dog-eared source deck and a few minutes came back with a brand new deck of cards, all the same color and all with typing on the top of them. She set them down, on edge, in a spare punched card box, making sure they were evenly aligned. She picked up a felt-tipped pen and, with a smooth, even stroke, drew a diagonal line from bottom left to top right across the top edges of the cards.

I'd seen that diagonal line before -- sort of a poor-mans way of keeping the cards in the right order. I knew that the black mark on the top of a card out of sequence would stand out even if only misplaced by one or two cards from its original order.

Maria then smiled and handed me the box with my cards in it. A real programmer's box! I think she knew how much that box meant to me.

No longer would I have to rely on a rubber band to hold the cards in my source deck together. I looked like a programmer as I nonchalantly carried my new cardboard box with my new source deck back to my office. I took the long way, stopping by my boss's office to ask him some innocuous question.

I was digging deeper and deeper into the details of my program, changing this and adding more there. Sometimes I punched my own cards, and sometimes Maria helped me create new cards for large sections of new program instructions. Soon I had filled half the box with punched cards.

As my program grew my confidence grew along with it. One day I was talking with the manager of the computer lab about how best to fit my growing program into the limited memory size of our computer. He had some suggestions that I hadn't thought of. When we were done he said: "I think you should have a key to the computer lab. Then you can go in and out whenever you want."

He gave me a key and left me to explore my newly authorized domain. I turned the key in the computer lab door and walked in.

I stopped just inside the door. I had been given the Secret Handshake, the Password, and the Magic Wand, all at once. I stood tall, surveying my domain.

Figure 5 IBM 360 Series 50

We had two computers, both inside this carefully guarded space. Desk-sized boxes scattered around the room held disk drives, printers and card readers. Tall refrigerator-sized magnetic tape drives completed the cluster of equipment surrounding each of the two computers. Along the wall to the left were some punched card machines similar to those Maria had in her room. One of the programmers sitting at a keypunch machine saw me, waved, and went back to his work I belonged.

Figure 6 IBM 2540 Card Reader/Punch

But two things immediately struck me. First, it was cold! A steady flow of cold air came from the ceiling vents. Computers didn't like heat, and the building air conditioner had been cranked all the way down. I learned later that if the temperature in the room got over eighty degrees the computers would shut themselves down. That was definitely not a good thing, since restarting each computer would take several hours. Next time I would bring a sweater.

Figure 7 IBM 1403 Chain Printer

And it was noisy. Through the open window to the computer lab I had seen a world of muted efficiency. Once inside, though, the noise was an enveloping cloak impossible to shed. A loud rumble came from the building air conditioner with its fans turned all the way up. Then came a higher whine from the ventilating fans inside each of the many desk-sized and refrigerator-sized boxes. Fan-fold paper flew out of the printer to fall in a neat pile, driven by silken motors almost inaudible under the sound from the whirling metal bicycle chain inside the printer, and the hammers needed to pound ink onto the paper could have been sound effects for that war movie I saw last night. Suddenly the punched card reader announced its presence when a stack of punched cards descended into the body of the reader, eaten by the whirring, clacking tin grasshopper hidden inside. Below these brash noises was a softer hum from one of the magnetic tape readers in the room, and a more erratic note from the disk drives that crowded around the computer. The computer itself was mute. Nothing moved on that magic box other than the flashing orange and yellow lights on the console.

Just then the computer operator went over to the ‘in’ counter and picked up the next request form and read it. He took the stack of punched cards that accompanied the request form and put them into the input hopper of the card reader. I watched the stack of punched cards disappear into the card reader to be chewed on by that tin grasshopper and reappear in a slot much lower on the machine. The operator restacked the punched cards in its original cardboard box and went over to the printer where he gathered up the stack of paper that came out of the printer. The paper and the punched cards went onto the ‘out’ section of the counter.

This is the cycle he repeated over and over during the day. Well, actually it was a bit more complicated than this. Three different program requests were active at a time; the request actually being run on the computer, the request that had just finished running, and the next one to be run. In addition, he replenished the paper in the printer as it was used up and made sure blank cards were in the cardpunch machine ready to be punched. He performed this complicated little dance, weaving around his dance floor with stately grace. I remember thinking that it took two to tango, and I didn’t know what dance would accommodate all of his partners.

The next day as I worked at my desk I had the source deck in front of me. I had pulled several cards out and was thinking about what corrections I needed to do, when somehow the cards slipped and ended up on the floor. Only a few, and I quickly put them back in the correct order, but the incident made me think.

Maria had made that wonderful diagonal mark across the top of the source deck but since then I had inserted quite a few cards that had no mark at all on the top. I had even moved a few cards around, making the diagonal line distorted at best. What would have happened if I had dumped the whole box? That would be a disaster. Yet I never heard of other programmers worrying about it. How did they deal with that potential problem?

“Bob,” I asked, “How do you keep your source cards in the proper order?”

He gave me that “You don’t know?” look and said: “How many columns are there on a punched card?”

“Eighty,” I said. “Everybody knows that.”

“OK,” Bob responded, “How many columns do you use to describe your program?”

“Seventy two. The compiler only looks at the first seventy two columns on each punched card.” I answered.

"What do you do with the remaining eight columns, columns seventy three through eighty?"

"Nothing."

"And, can you think of anything that you could use those columns for?"

I thought for a minute. "A sequence number?"

He smiled and turned back to his work.

We had a punched card sorting machine that I had used a few times before. Another desk-sized machine, it was a vital part of the punched card process. While it would only sort one card column, or one digit of a number, at a time it would only take a few minutes to sort using the eight-digit sequence number, if necessary.

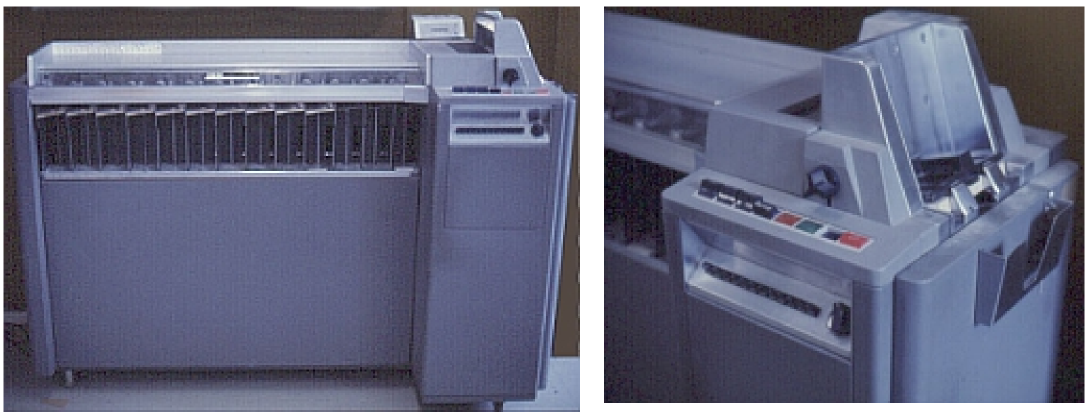

Figure 8 Punched Card Sorter. The knob in the center of the right photograph is the dial that selects a specific column to be sorted.

Ten minutes later I was back in his office. "I've decided that it's too much work. Whenever I add a new instruction to the program I'm out of sequence. And when I need to reorder the instructions somehow, the sequence numbers will be all wrong anyway. I'll take my chances."

He looked at me soberly, thought for a moment and said, "Let me tell you a story. Do you remember Andy?"

"Yes." I said. Andy had moved on several months ago. He was a good programmer but self-centered and not well liked by the other programmers.

"Andy didn't believe in sequence numbers," Bob said. "And I worried about that, especially since he would stack his source deck into the card reader without worrying about who was ahead of him. Then he would leave, and when the computer had run his program he wouldn't be there to take care of his punched cards. It was up to the rest of us to take his cards out of the card reader and stack them somewhere until he finally came back to get them.

“One day I decided to teach him a lesson. He had left his source deck in the card reader, as he usually did, and when his program finished running Andy was nowhere around. I took his cards out of the card reader and set them on the floor. Then I got another stack of blank cards about the same height as his and put that stack on top of the card reader, right where we usually put his source deck.

“Another guy and I waited for Andy to come back. When he did, I said to the other guy ‘And it was THIS big!’ spreading my arms wide and knocking the stack of blank cards all over the place.

“Andy looked at me, looked at the cards all over the floor, and said ‘Oh shucky-darn’ or something to that effect. He then pulled back his right foot, and he gave that neat stack of cards on the floor a mighty kick!”

He got a faraway look on his face for a moment, and little laugh wrinkles appeared around his eyes. Then he looked at me thoughtfully and said “We have a program that will read a source deck and punch out a new copy with new sequence numbers on them.”

From then on I used sequence numbers.

My program grew. My cardboard box was almost full and it dawned on me that other people were asking me how to solve little programming problems. I spent less and less time in Bob’s office, and the questions we discussed became more about the philosophy of programming and less about the technical issues of how to write a program.

One Saturday I went to the Lab to work on my program. I was a little behind schedule and wanted to have the computer to myself. Or so I thought! I recognized several cars in the parking lot as belonging to other programmers, and when I opened the door to the computer lab they were clustered around the computer. Past experience told me that was a bad sign. Was the computer broken?

The stillness of the group was ominous. Only two of them moved; the rest were intently watching them. Bob was reading aloud from an instruction book while one of the other programmers typed on the computer typewriter.

“We have it!” The computer operator said to me softly, not wanting to disturb Bob.

“What do we have?” I asked fearfully.

“We have the new MFT operating system. From now on, we can run more than one program at a time on our computer. At least we should be able to Monday morning!”

They were installing a new operating system for the computer! This would take a while, but at least nothing was wrong with the computer. That was the good news.

I went home, disappointed at the lost time. I wouldn't be able to do anything more until Monday. All for a programmer's toy that wouldn't help me at all. Why would anybody care whether a computer ran two programs at the same time? Each program would only run half as fast, wouldn't it? The computer only had so much work it could do, so many instructions per second. Or minute. Or hour. At the end of the day no more work would have been done, right?

I didn't realize then that this new operating system would forever change how programming was done. Within a month I had set aside my punched cards, and Maria's career as a keypunch operator was over.

Remember that second computer? I never used it. Neither did any of the other programmers. The only people who used that computer were working on a project to support document creation. I don't remember the name of the project, but the secretaries in the lab loved it.

Each of the secretaries had been given a Selectric typewriter just like the one that the computer operator had. Those typewriters were connected to that second computer, and the secretaries could save and edit long documents, something they had never been able to do before! No more carbon paper and no more whiteout smears!

I got involved because I was writing, with our secretary's help, a long technical paper describing our project objectives. I remember that typing the cryptic characters p/322 (p for print) would cause the typewriter to print line 322 of the report:

Merry had a little lamb

Then, typing c/Merry/Mary/ (c for change) would cause the revised line to be printed out:

Mary had a little lamb

It was wonderful. I loved it and probably spent as much time as our secretary working on my document. I think I must have been a pest. When I was at work I would bounce back and forth between working on my program and writing the new report for our project. I finally agreed that I would use the typewriter only during the lunch hour.

The program that talked to all those Selectric typewriters ran on that second computer. There it ran all day, changing things here for one secretary and printing things there for another. But the computer was essentially useless to the rest of us programmers – it just sat there. No big reports were printed and no decks of punched cards came or went. The computer operator's dance never involved that second computer. All the activity occurred at the typewriters scattered around the Lab.

But when the new operating system, MFT, was installed, this document creation program was moved to 'my' computer. I grumbled, because it meant that all the rest of the programs, including mine, ran a slower. Besides, MFT took twice the amount of core memory, 64 kilobytes, meaning there was that much less memory for my program. The computer only had about 512 kilobytes, so the amount of memory space left over was reduced significantly. This was progress?

I would work on my program during the morning, and when our secretary went to lunch I would race over to our Selectric typewriter to work on my report. When she came back from lunch I would reluctantly give it up and go for lunch myself.

Bob stopped by one day on his way to lunch. He hadn't spent any time with the document creation program before, and this time I became the expert, showing what the program could do. It made me feel good to answer his questions for a change! He didn't say much, but I got the feeling that he was impressed.

A couple of days later he came back. He said: "We've figured out a way to make the document creation program look like the punched card reader to the computer. We need to make a copy of your source code deck and store it just like the report you are working on, and then you can use the Selectric typewriter to make your program changes. We need you because you already know how to use the typewriter. I'll help you get everything organized the way you will need it.

"Let me borrow your source deck. I'll bring it back in a few minutes," he said.

Bob was one of few people I trusted with my source deck.

He came back shortly, and I put my source deck back in that special place on the shelf where I kept it. Then Bob and I worked together at the typewriter - long past the time that the secretary came back from lunch - and finally we had it. The computer was running my program! We went down to the computer lab and from the 'out' counter picked up a stack of computer printout. No source deck and no program deck. The information on both had been saved on the disk storage attached to the computer. From now on, any time I wanted I could use the document creation program to edit what I used to call my source deck, or I could ask the computer to run my program. All I had to do is sit down at the typewriter and type in a command to the computer!

But this would take some getting used to. For one thing, I had second priority on the secretary's typewriter, a distinct problem. Could I get a typewriter?

"No,' Bob said, "But we will set one up in a room down the hall that we can all share."

The next several days were hectic. All the programmers were learning how to use the document creation program, and there was always a line of people waiting to use the Selectric typewriter down the hall.

A few of the programmers were reluctant to make the change. Giving up their punched cards was a big leap; being able to hold their source deck in our hands was important. Also, the ability to hand a coding sheet to Maria in the keypunch room had always been heavily used by those without typing skills. But soon even the non-typists were converted, however reluctantly, to the typewriter. The increased productivity, even for a hunt-and-peck typist, was just too great to ignore.

My source deck stayed on the shelf where I had set it, and the card reader in the computer lab now sat idle most of the day. The computer hummed, though, and while the ‘in’ counter grew dust bunnies, piles of computer printout grew on the ‘out’ counter. It no longer held stacks and stacks of punched cards.

One evening, as I straightened up my office getting ready to go home, I thought about that card deck sitting on the shelf. I hadn’t used it for several days, and, it was out of date. I had used the document creation program to make several changes, some important, to my program without bothering to update the punched card deck. Clearly some major effort would be required to update the card deck, and if I didn’t do that, the card deck would shortly be worthless.

However, all that effort would be wasted if I didn’t use the card deck.

Would I use the card deck again? I couldn’t think of a reason, as long as the document creation program continued to work as it was. Bob had assured me that I could, if necessary, ask the computer to punch out a new source deck if I needed it. Reluctantly I decided that I would not update my source deck.

Pulling the box down from its shelf I lifted the top and looked at the cards. The box was now almost full. Remnants of the diagonal stripe that Maria had drawn long ago on the top of the cards were still there, interrupted by cards I had inserted later. There were some finger smudges here and there, and on the left side of the cards was a brown smear where I had slopped, and then quickly wiped away, coffee one day.

I ran my hand over the top of the cards and memories floated up, as if the cards were speaking to me in Braille. The long evening I had spent creating a small section of bright blue cards, just a hair bigger than the surrounding cream cards. The difficult debugging process that went into a particularly uneven section of cards, where I had replaced and revised many cards before I got it right. My fingers gliding over a smooth section reminded me of the satisfaction and pride I had when the cards Maria created for me were intact, having required no debugging at all.

Figure 9 IBM 2311 Disk Storage Unit

Throwing away that card deck would be throwing away a part of my life.

I got up and walked down to the computer lab. Inside, the familiar cold air and muted roar welcomed me as I walked over to the disk storage units that now held my program source – information? I chuckled to myself, thinking that I could no longer call it a source deck, and that I didn't know what the new name should be.

We had four of the disk storage units, each holding well over seven million characters of information. One of the four had my source data. I had earlier decided that as long as the computer knew where it was, and could connect me with it whenever I wanted it, I didn't need to know exactly which one.

I could see a stack of spinning disks through the clear Plexiglas cover, and I knew that little tiny magnets embedded in the thin coating on those spinning disks now defined my data. This was my future, whether I wanted it or not. As I had done with my card deck, I reached out and put my hand on one of the disk storage units, but the machine didn't speak to me. That faint impersonal vibration had no meaning, invoked no memories.

Back in my office I closed the lid on my box of punched cards and carefully put it back on the shelf in its familiar place.

The next day, feeling like a pallbearer at my own funeral, I brought my source deck down to the computer lab and laid it tenderly in the recycling bin there. And then went back to my empty office.

We got several more typewriters, and these were immediately put into use.

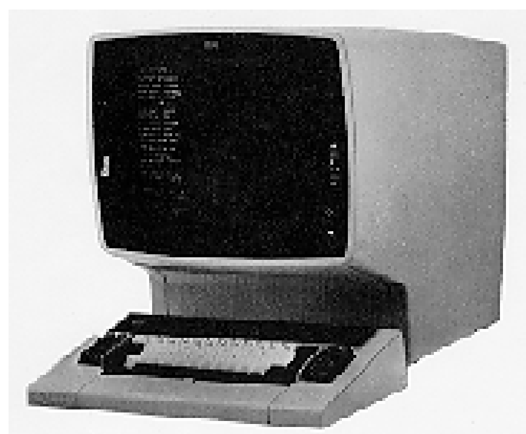

Figure 10 IBM 3270 Display Terminal

A shipment of television-like display terminals arrived, and they were rapidly set up around the building to replace and augment the typewriters. The computer lab process of managing a bunch of programs, all submitted by different programmers at the same time, became more streamlined, and the turmoil in our daily schedule slowly died back to its normal level of chaos.

I finished my program. At least as much as any useful program is ever finished, as I continued to receive requests for improvements. I enjoyed the feeling of accomplishment when the program became more and more useful; the feeling of satisfaction when others used my program.

Maria stopped by my office one day. "It's been great working with you," she said. "I'm not sure where I'll be going."   We walked down to the cafeteria where I bought her a cup of coffee, and we chuckled remembering my early attempts to master the art of punched cards. She showed me pictures of her family. A shadow lay over the day, darkening my mood as I walked back to my office alone. I knew that someone special had left of my life. I never saw her again.

A couple months later it dawned on me that she and I had shared a unique moment in history. There are words for it – an inflection point, a paradigm shift, a sea change. Whatever you call it, our world had changed forever.

A programmer's office was no longer filled with stacks and stacks of punched cards. No less messy, maybe, but no punched cards. Now I started to see reels of magnetic tape and, sometimes, magnetic disk cartridges, the very expensive spare-tire-sized floppy disk of the day. But mostly our programming world revolved around stacks of computer printout paper. Our shelves became repositories of paper.

No longer could we hold our program in our hands, feel its heft, measure our productivity by the pound. Now our programs were 'out there' somewhere, just as useful as before but not visible in the way we were used to. The electronic age hadn't mutilated that sturdy punched card, but it had figured out how to ignore it.

The Twentieth Century had, as a technical foundation, that little piece of cardboard with holes in it. The 1890 census started it, Social Security in the 1930's cemented it into our culture, and throughout the first half of the century IBM Unit Record Equipment supported and automated the growth of business and our society.

While we will talk about punching in and out of work for a long time, today that punched card is seen primarily on Election Day. The threat of 'hanging chads' will soon eliminate it even there.

Incidentally, have you ever heard the term 'Throw it in the bit bucket?' Computer people use it to imply that something has been deleted from the computer memory. The bit bucket has become a slang term for a computer trashcan, when those little tiny magnets on a spinning disk are used for something else. There really was a bit bucket. Whenever a card was punched, a small number of little, tiny rectangles of cardboard were punched out of the card. The chad. That chad went into a little tin container and was later discarded. That container was - guess what – the bit bucket.

The things in our lives go through a cycle. They start out being modern, then mature, dated, outmoded, obsolete, junk, finally retro and ultimately antique. The punched card is now approaching the antique stage.

My daughter still talks wistfully about that Christmas wreath we made by stapling blue punched cards together in a ring. One year a squirrel got into our attic and ruined it, and we can't duplicate or repair it – we have no cards.

Maybe we can find some on ebay.

Figure source web sites.

I am grateful for their visual contributions to this tale. The pictures are included for illustration only, and I do not hold or claim copyright on the pictures, which are the property of their respective owners.

Figure 1 http://www.nfrpartners.com/comphistory/punchcards1.htm

Figure 2 http://www.columbia.edu/acis/history/029.html.

Figure 3 http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_2423PH2040.html

Figure 4 http://www.columbia.edu/acis/history/029.html

Figure 5 http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_2423PH2050.html

Figure 6 http://www.columbia.edu/acis/history/2540.html

Figure 7 http://www.rwc.uc.edu/koehler/cshist.html

Figure 8 http://www.rwc.uc.edu/koehler/cshist.html

Figure 9 http://ibm1130.org/hw/disk

Figure 10 http://www.cs.utk.edu/~shuford/terminal/ibm.html