University of Waterloo
Red Room Stories

When the Mathematics and Computer Science building was designed, space was set aside for an ambitious computer room. It was two stories high, took up about half of the area of the first two floors, and (for reasons I've never understood) was painted this almost violent shade of red. Whence the name, the Red Room.

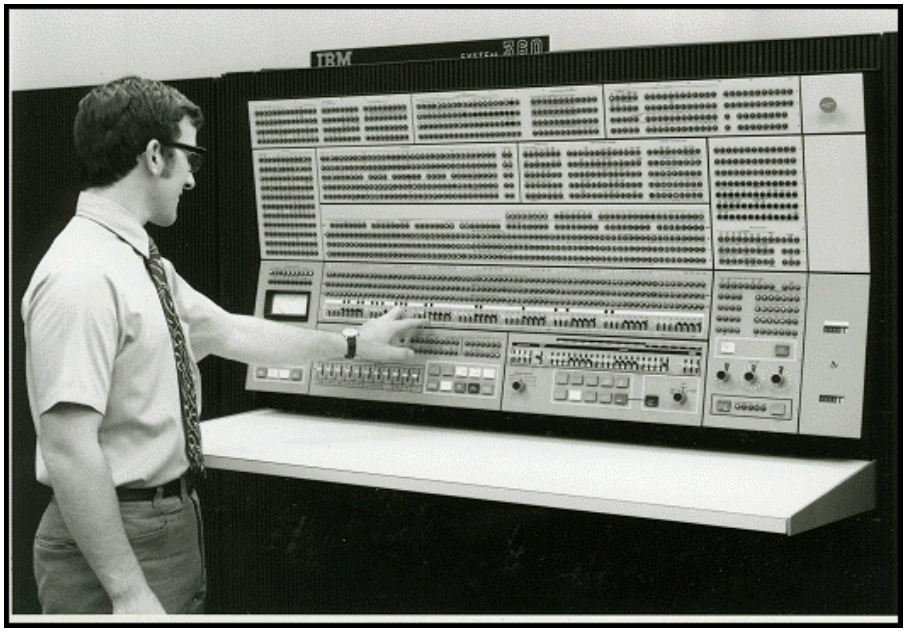
The centerpiece of the room was the System 360 Model 75, then one of the largest and fastest computers in Canada.

That's the main console of the 75 just to the left of center above. While computers today are small enough to carry under your arm (and far, far faster), back then this computer consisted of the console, the black "spine" leading to the right, and the 5 refrigerator+ sized appendages hanging off the spine. Peripherals such as disk drives (they're on the far right) were separate refrigerator-sized boxes. See later.

[A few technical details. The computer on my desk at the moment is 12" x 11" x 3.5", contains 8GB of RAM, has 4 CPUs (cores) inside, and runs at 3.2 GHz and cost around $1,000. Whereas the 75 was huge, held 1MB of RAM (upgraded from its initial 256KB, then 512KB and ultimately to 3MB), had a single CPU, and ran at 1.3 MHz. And cost somewhere around one million dollars (1967 dollars, no less!). You do the math.]

Note the area immediately to the right of the console, all black. That will be the "closet" in one of the stories below.

Here's a close-up of the console.



Note on the picture above. Except for hardware diagnostics by IBM support staff, and a few buttons to boot the system and such, the console was useless. (I actually saw it in use exactly once to debug a *very* obscure problem.) And the guy in the picture above, Bob Cressman, was a computer operator, which wasn't anywhere near as technical a job as you might expect. As such he would have known nothing about what the console lights meant. The only reason he has his fingers on some of the switches is that the photographer undoubtedly felt the picture was too static otherwise.

As mentioned above, the room was two stories high, surrounded on all four sides by Plexiglas, so that people could look down into the room. It was meant to be quite the showcase.

The following picture shows just a bit of the second floor.

[While we're here, those are a bank of 8 model 2314 disk drives (the 9th was a spare) with its Control Unit in blue to the right. Each disk (you can see the canisters that held them resting above the units) held a then massive 25 MB of data. Yes, you read that right. MB, not GB.]

A bit more background.

The University of Waterloo was then a very new university, barely 10 years old. As such, it was aggressively promoting itself. The showcase aspect of the second floor Plexiglas was quite deliberate, especially during Computer Science Days.

There were no such things as personal computers back then, and high schools with even a small computer or even remote access to one were very few and far between. So the Math Faculty came up with the idea of Computer Science Days. Each week during the school year, half a dozen high schools from around Ontario were invited to send 30-40 students interested in learning computer programming up to UoW. And so the school buses arrived.

The students were given introductory programming courses and allowed to run any programs they were then capable of. There were university students hired as "Demonstrators" to answer the kids' questions, help them write and debug their programs, and so on. A grand time was had by all.

Translation: Please enroll here when you've graduated from high school!

I have several stories from my days at UoW back then (1967-1972). Here are three.

Coming Out of the Closet

I was one of the Computer Science Day demonstrators from (roughly) 1967-1972. I also worked part time for the Computing Centre.

In the earliest days, Computer Science Day high school students were actually allowed inside the red room (with strict orders not to touch anything).

Later (and more sanely), they would be told to gather on the second floor, looking down at the console of the 75. One of the computing centre staff would use a microphone (and very long cord) that was attached to speakers on the second floor. He would walk around the red room. "Here's the main computer console. Over here are the disk drives." And so on.

We never did the following. To this day, I regret this. What I had in mind was this…

The chassis of the 75 not only was enormous, but you could open panels and find open space (so that IBM Customer Engineers could access the circuit boards there). In some cases you could even walk in, turn around, and close the panel behind you. One of these "closets" was immediately to the right of the 75 console.

One of the other demonstrators was Andy Lawrence. On the day I came up with this idea, Mike Doyle (head of system programming, and a good guy with a sense of humor) was to be the

person with the microphone doing the tour. A few minutes before he was to start, unknown to him I would hide inside the "closet" with a pad of paper and a pen.

Mike would start doing his "here's this ... here's that" spiel. Andy would be offstage, in the area just off the red room. As Mike was doing his "here's the main console" bit, Andy would enter the red room, go over to Mike, nod to him, smile, then hit the STOP button on the console. At that point, the always blinking lights would FREEZE! He would then go over to the access panel of my closet and knock. I would open the panel, come out, nod to him, and pass him my pad of paper and pen. He would then enter the closet and I would close the door behind him. On my way out I would press the START button, the lights would start flashing again, and I would sail out, stage left.

I can only imagine Mike's reaction to this. But it would have been soooo funny!

Label Screening

In the summer of 1968, I was a student working for the summer at the Computing Centre.

There were four of us, including Vic Neglia (still working for ArtsFac, as I understand). (And was Jim Welch on the team as well? I think so.)

Our project was Genasys -- General Addressing System.

The basis for this was the fact that each departmental secretary had several mailing lists. Perhaps one for professors, another for alumni, etc. Genasys would allow them (after initially entering in the information in each list) to select a list (say, ALUMNI) and print out mailing labels, one for each recipient. They could also select more than one recipient list, and we'd print out labels for each name on the list, suppressing duplicates.

The project was strange in the sense that while ultimately the goal was to have typewriter terminals in each department to print the labels, our development environment was on 2260 CRTs. We were "printing" labels on a screen! Not too useful. And ultimately, the program fizzled away.

But until then, there was a story.

First a bit of background on the operating system we were using. IBM's most advanced operating system at the time was called MFT (Multiprogramming with a Fixed Number of Tasks). This was later superseded by MFT-II and MVT (Multiprogramming with a Variable Number of Tasks). But we were stuck with MFT, which had a number of non-trivial limitations. The relevant one here was as follows. When you installed the operating system, you had to specify the number of programs you could run concurrently. Partition 0 (P0) was the highest priority, followed by P1 and so on down to (I think) P6. And these had to be run *in order*.

So when the system was booted, the operator had to run an installation-specified program in P0 (it turned out to be Genasys). Then once it started, run a specified program in P1, and so on. And if any program crashed, you couldn't run a program in that partition again until all the lower

numbered partitions were finished. So the upshot was that you had only the lowest priority partition (P6) in which to run jobs. So it was multiprogramming only in a very limited sense. Essentially you could run only a single job at a time (in P6), plus the background programs running in P0, P1, etc.

It cramped our style while developing Genasys. If we made a coding error and Genasys crashed, we essentially had to wait for the system to be rebooted before we could run it again. It just wasn't in the cards to stop running normal programs in P6, just so we could shut down P5, P4, P3, P2 and P1, just so we could restart Genasys in P0, and then start programs in P1 ... P6 again.

We finally got to the point where we had enough working to give a demo to Don Cowan (the head of the computer science department). He was due at 9:30 AM. Most of the developers got there early for last minute testing. And we crashed it!. One of the guys had made a change the previous night to the mailing list merge routine. It worked fine if you gave it two or more lists, but crashed if you gave it only one list.

So I was put in the embarrassing situation of going to Gwen, the shift supervisor, and telling her we needed her to reboot the 75, because we had to give Don Cowan a demo in an hour. Well, she wasn't pleased with this. At all. But she gave in. Thank you Gwen.

So the system was rebooted, and Genasys was alive again. We had half-an-hour or so to kill, so we went to our offices for a while, or went to the third floor to get a Coke, or whatever. When we came back, we found that the final member of the team (Jim Welch, I think) had come in and decided to test out the system. So, not knowing what had come before, he typed in a request for a single mailing list. CRASH!

There was absolutely no way I could go back to Gwen and ask her to reboot. Again.

There was only one thing to do (no, I didn't even *think* about postponing the demo). MFT was pretty unstable. There were any number of ways to write a program that would crash it. So for the only time in my life, I (deliberately) crashed the mainframe.

And that brought Genasys up again.

There's a follow-on to this.

When Don Cowan came in for the demo, I was the one at the keyboard, with the rest of the team standing behind Don. I explained to him what he was seeing, what the user interface was and so on. Fine. "OK," he said, "let's see it process mailing list ABC." I didn't hear, but I certainly sensed, all the other members of the team gasping. "Sure, Dr. Cowan. But we have this nice merge facility. Let's process lists ABC and DEF". And everyone exhaled.

The demo was a success. But as I said, the project itself failed.

C'est la vie.

For Whom the Bell T-t-t-tolls

While in later days, the 75 used CRTs as consoles, originally it came equipped with a 1052 typewriter console (basically an IBM Selectric, with a golf ball mechanism). The console also had a bell that sounded like a bell a kid might put on his bike. This was used exclusively by the operating system (MVT) when it crashed (a disappointingly frequent occurrence in those days). As a last gasp, it would (if it still could) ring the console bell.

While there was a call to MVT that would let you type a message on the console, there was no interface that would let you ring the bell. Even if you tried to access the device directly (as you could for disks, tapes, etc), you'd fail. The operating system had it allocated, and no others were allowed to send commands directly to it.

Around 1969 or so, I figured out how to bypass MVT's security (such as it was in those early days) and get access directly to the console. At that point I could actually ring the bell.

I was a student, but worked part time for the Computing Centre, so I had ready access to the 75 after hours. When I got my program working, I was a bit disappointed. The bell rang, but only for a second or so, and I wanted it to ring for perhaps 5 seconds.

So I modified my program and sent the command to ring the bell 3 times. Surprisingly, it still rang for only a second. I modified it again to ring it 5 times. No change. 10 times. Uh, no real change. 20 times. Now I started getting something. But instead of coming out as a steady ringing tone, it sort of went brinnng-brinng-brng-... Ah! The bell was implemented in terms of a capacitor, and calling it repeatedly didn't allow it time to recharge. OK, so I set the repetition factor to something like 30, and got my 5 seconds out (albeit in a stuttering fashion).

Fine.

A day or two later was the day. April Fools day, in fact.

So about 2 in the afternoon, I submitted my card deck. I'd added an extra line to the program to type "APRIL FOOLS" on the console when the program ended. I then went up to the second floor to watch.

Sure enough, in 5 or 10 minutes, the bell rang. I remember seeing the look on the operator's (Nancy's) face. Shit! The system just crashed. Again.

So she started to reach for the pad of paper with the shift report on it. When the bell rang again! Her head whipped around. Huh?!?

Then it rang again. Pure puzzlement on her face.

Then it rang again. And again. And again.

Everybody was totally confused. The system programming staff were summoned.

My testing had been in essentially a standalone environment. But in the middle of the afternoon, the console was constantly busy with system messages. Such-and-such job was starting. Such-and-such job had ended. Please mount a tape on drive 0C0. And so on.

So what happened was my program rang the bell the first time, but before I could send my command to ring it again, the system needed to type a message. Then the bell would ring again. Then another system message. For about 30 times.

Then the system hung and they did have to reboot (or, as we called it in those days, re-IPL (Initial Program Load)).

Word got out about ringing the bell (all the students on the second floor next to the red room could hear the bell and realized something strange was happening). Shortly thereafter, I ran across my friend Peter Wooster and it came out that I was the one who had done it. He got excited. His co-op job was with Air Canada in Winnipeg, and they had an outstanding bet that no one could ring the console bell. So he wanted a copy of my program.

Well, sure. So I went to the second floor card reader / printer terminal and was in line to list my card deck for him. As I was standing there, who should happen to come in but Mike Doyle, the Manager of System Programming, also wanting a listing! Omighod! As I tried to shrink into nothingness so that he wouldn't see me, I had vivid images of his bawling me out, but good, not just for the disruption, but for actually crashing the system.

And sure enough, the first words he said to me were, “I see you rang the bell.”

“Uh, yeah.”

“But it’s too bad your APRIL FOOLS message didn’t come out.”

And the strange thing was that there was a smile on his face. It dawned on me that he was taking the whole thing as a good April Fools prank. I might actually survive this!

He then told me that the reason the program crashed the system was that when I modified system memory to get around MVT's security, I didn't restore the data before exiting my program. And that confused MVT and led to the crash.

And how did he know it was me, and how did he know what the programming was? Because when the system came back up, and my program (with source) printed out, my userid was on the JOB card. And how did he know to look at that particular job? It probably had something to do with the fact that I'd called the job DINGDONG!

Larry Smith

May 2012